LEARNING TO MANIPULATE LATENT REPRESENTATIONS OF DEEP GENERATIVE MODELS

towards improving interactivity and controllability in automatic music creation

A Dissertation
Presented to
The Academic Faculty

By

Kumar Ashis Pati

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Music

Georgia Institute of Technology

May 2021

LEARNING TO MANIPULATE LATENT REPRESENTATIONS OF DEEP GENERATIVE MODELS

Approved by:

Dr. Alexander Lerch
School of Music
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Gil Weinberg
School of Music
Georgia Institute of Technology

Dr. Jason Freeman
School of Music
Georgia Institute of Technology

Dr. Mark Riedl
School of Interactive Computing
Georgia Institute of Technology

Dr. Rudolph van der Merwe
Advanced Computations Group
Apple, Inc.

Date Approved: Dec 11, 2020

ACKNOWLEDGEMENTS

This thesis is a culmination of years of work at the Georgia Tech Center for Music Technology (GTCMT) and would simply have not been possible without the support and assistance of several people. I would like to take this opportunity to extend my sincere gratitude towards them.

First and foremost, I would like to thank my advisor, Prof. Alexander Lerch, who has been instrumental in shaping and guiding my research over the course of my graduate studies. The confidence that he showed in me during my early days at Georgia Tech is what strengthened my resolve to pursue a Doctoral degree. His constructive and insightful feedback on my work has helped me polish my critical thinking skills and improve the overall rigor of my research. I am confident that our association will continue to grow over the coming years.

I would also like to thank all the members of my thesis committee: Prof. Devi Parikh, Prof. Gil Weinberg, Prof. Jason Freeman, Prof. Mark Riedl, and Rudolph van der Merwe. Their feedback on different aspects of my thesis proposal and during the defense has helped me in improving my thesis substantially. In particular, this has helped me contextualize the thesis in a better manner within the music and machine learning domains.

Deciding to pursue graduate studies in Music Technology five years ago was a difficult decision because of the major shift of focus it was about to have on my career. This decision would not have been possible without the guidance from Sankalp Gulati, who has ever been a constant source of inspiration for me.

The work presented in this thesis has relied heavily on the support and assistance of all my collaborators with whom I have had the pleasure of working. I would especially like to thank Gaëtan Hadjeres, and Siddharth Gururani for helping me out with the different publications which form an integral part of this thesis. I would also like to extend my gratitude to my colleagues from the Music Team at Sony CSL and the Advanced Computations Group at

Apple during my internship days. I cherish the several discussions I have had with them on various topics not only related to this thesis but also on music informatics and applied machine learning. In addition, I would like to thank Siddharth, Amruta, Chih-Wei, and Vinod for providing useful feedback during the course of writing this thesis which has allowed me to fine-tune it even further.

Graduate student life in a foreign country requires a lot of support outside of academics. In particular, I am grateful to my roommates Ankit, Anuj, and Mohit. Life in Atlanta would have not been the same without them. I would also like to thank the GTCMT graduate student cohort and faculty for helping me find my footing in the new academic environment during those initial days at Georgia Tech.

Last but not the least, this thesis would not have been possible without the constant support and encouragement from my parents and my sister. Their unconditional love and unwavering belief in me have allowed me to freely pursue my interests which has led me to this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Deep generative models have emerged as a tool of choice for the design of automatic music composition systems. While these models are capable of learning complex representations from data, a limitation of many of these models is that they allow little to no control over the generated music. Latent representation-based models, such as Variational Auto-Encoders, have the potential to alleviate this limitation as they are able to encode hidden attributes of the data in a low-dimensional latent space. However, the encoded attributes are often not interpretable and cannot be explicitly controlled.

The work presented in this thesis seeks to address these challenges by learning to manipulate and design latent spaces in a way that allows control over musically meaningful attributes that are understandable by humans. This in turn can allow explicit control of such attributes during the generation process and help users realize their compositional goals. Specifically, three different approaches are proposed to investigate this problem. The first approach shows that we can learn to traverse latent spaces of generative models to perform complex interactive music composition tasks. The second approach uses a novel latent space regularization technique which can encode individual musical attributes along specific dimensions of the latent space. The third approach attempts to use attribute-informed non-linear transformations over an existing latent space such that the transformed latent space allows controllable generation of data. In addition, the problem of disentanglement learning in the context of symbolic music is investigated systematically by proposing a tailor-made dataset for the task and evaluating the performance of several different methods for unsupervised and supervised disentanglement learning. Together, the proposed methods

will help address critical shortcomings of deep music generative models and pave the path towards intuitive interfaces which can be used by humans in real compositional settings.

CHAPTER 1
INTRODUCTION

Creativity is considered as one of the core components of general human intelligence [1]. Since the development of computers, there has been considerable interest in building *creative* computational systems. This field of research is called Computational Creativity (CC) and is defined as [2]:

> *The philosophy, science, and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviors that unbiased observers would deem to be creative.*

Even though there is some ambiguity regarding the precise definition of *creativity* [3], simulating creative behavior in machines can be considered as one of the major challenges in the field of Artificial Intelligence (AI) [2]. Consequently, the quest to build creative computational systems has spanned several diverse fields, including, but not limited to, mathematics and science, logic, industrial design, coding, story-telling, poetry, music composition, and performance [4].

Music, in particular, is interesting on account of its inherent ability to cause strong affective response [5] transcending social and cultural boundaries [6, 7]. Music is considered as one of the primary forms of social engagement and entertainment across different cultures. Talented music composers are often regarded as highly creative individuals. Thus, computational systems for music creation are of significant interest to the CC research community. This is also evidenced by the high number of contributions in topics related to music generation in top conferences in CC [4]. In addition to this interest from the research community, computational systems for music generation have the potential to change the way music is created and could enable a wide range of potential applications which are

briefly discussed in Section 1.1.

Consequently, there has been significant research on building automatic music composition systems using different methods and approaches. Section 1.2 presents a brief overview of the history of the evolution and current status of such systems. In recent years, machine learning techniques have emerged as the tool of choice for the design of automatic music generation systems [8] with Deep Learning (DL) [9] being the most widely used [10]. While many of these *deep generative models* have been successfully applied to several different music generation tasks, e.g., monophonic music generation consisting of a single melodic line [11, 12, 13], polyphonic music generation involving several different parts or instruments [14, 15], and creating musical renditions with expressive timing and dynamics [16, 17], they are often found lacking in two critical aspects: *control* and *interactivity* [18]. Most of the models typically work as black-boxes, i.e, the intended end-user has little to no control over the generation process. Additionally, they do not allow any modes for interaction, i.e., the user cannot selectively modify the generated music or some of its parts based on desired musical characteristics. These challenges are discussed further in Section 1.3.

This thesis presents several methods to address the interactivity and controllability challenges associated with deep music generative models. The primary motivation for the proposed methods stems from the field of representation learning [19] which deals with learning meaningful low-dimensional representations from high-dimensional data. These so-called *latent* representations, described further in Section 1.4, are able to encode certain hidden attributes or properties of the data [20]. The proposed methods and techniques in this thesis will seek to leverage and manipulate these latent representations in order to improve the interactivity and controllability of deep music generative models. In addition, the proposed methods and techniques are developed with a general machine learning framework in mind, and hence, can be extended to other domains in the field of CC at large. This is shown by applying the proposed methods to image generation tasks as well.

1.1 Applications of Automatic Music Creation Systems

Developments in automatic music creation systems enable a wide range of applications which have the potential to change the way we create music and interact with it. Some of the different areas where such systems can be useful are discussed below.

(a) **Creating Better Compositional Aids**: The most obvious application area of automatic music creation systems is in providing music composers and creators with better compositional tools (e.g., Google's Magenta Studio[1]). Such tools are typically designed to improve or simplify different aspects of the music composition process. Some of the areas where these tools could be extremely useful are: harmonizing melodies [21, 22, 23], suggesting musical arrangement, orchestrations, and accompanying instruments such as drums and bass [24, 25, 26, 27]. They can also be used to obtain new compositional ideas by either suggesting new musical material or suggesting continuations to existing musical ideas [16].

In addition to the above, ongoing research on musical style transfer [28] has also opened up possibilities for several interesting applications. An automatic tool which can transform music by changing its orchestration or arrangement to different genres would be extremely useful for a composer by allowing them to try different ideas quickly. Such tools might also find traction amongst Disc Jockeys looking to mash-up songs with different styles.

For all these compositional tools to be useful, they need to provide the users with the ability to control the generated music by allowing meaningful opportunities for interaction.

(b) **Enabling Music Creation for Media Applications**: The rise of internet and social media has created a huge demand for new media content. Video streaming services,

[1] https://magenta.tensorflow.org/studio, last accessed: 26th October 2020

such as YouTube[2], have resulted in a large number of videos being produced and consumed. However, the content creators of these videos might not always have the financial means to license commercial music or the necessary musical expertise to compose and create their own music for the video soundtrack. This high demand for original music for video content has led to the creation of several start-ups (e.g., Amper Music,[3] and Aiva[4]). These are focused on composing music based on either the video content or certain high-level control parameters such as the emotion or mood in the video.

Another area which could benefit immensely from automatic music creation systems is the console and online video games industry [29]. Video games often rely on using elaborate and compelling soundtracks to improve the gaming experience. Studies have also shown that video game soundtracks lead to greater enjoyment and stronger affective response in players [30]. Building music generation tools that can consider the game scenario, as well as the players' actions as inputs to the compositional process and the sound design, could help accentuate the gaming experience even further.

(c) **Democratizing Music Creation**: Over the years, different technological developments such as Digital Audio Workstations (DAWs) have made music creation more accessible to a wider audience. The availability of online tutorials and applications for music education (e.g, Coursera[5]) has allowed people to learn about music theory. Developments in computational music creation would be another step towards the democratization of music creation. Content creators (especially with limited musical background and experience) would have access to simple, easy-to-use tools which would enable them to explore music creation in novel ways. For instance, tools such

[2]https://www.youtube.com, last accessed: 26th October 2020
[3]https://www.welcome.ai/amper-music, last accessed: 26th October 2020
[4]https://www.aiva.ai, last accessed: 26th October 2020
[5]https://www.coursera.org, last accessed: 26th October 2020

as the piano genie controller [31] can allow non-musicians to improvise and perform on different musical instruments using basic controls such as the overall melodic contour.

(d) **Aiding Music Information Retrieval Research**: Music Information Retrieval (MIR) is an area of research that deals with computational analysis of music data and automatically extracting any useful information therein [32, 33]. The field of MIR over the years has gradually transitioned from feature design to feature learning-based methods [34]. This transition has been caused due to an increasing reliance on DL-based methods which are known to be extremely data-hungry. While this has resulted in significant improvements in performance across many different MIR tasks, it has also resulted in an increasing need for large annotated datasets which are difficult and costly to obtain in most cases [35, 36]. One of the possible approaches to overcome this limitation has been to use artificially generated data [37]. Improvements in music generation models can help MIR research by artificially creating large quantities of annotated music data. This would directly benefit several different areas within MIR research such as source separation [38], music transcription [39], and music performance analysis [40]. The ability to control specific aspects of this generated data will be important especially for supervised and semi-supervised learning methods.

(e) **Other Applications**: There are also some other areas which would benefit from improvements in music generation systems. One such area is music therapy [41], where generating music with specific properties and target emotions [42, 43] would help accomplish therapeutic tasks such as improving relaxation and mindfulness [44]. Another application area would be personalized music streaming. Context-aware music generation systems can be used to connect recommended songs in interesting ways which have the potential to create new listening experiences.

1.2 Brief History of Computer Music Composition

Computer music composition or computer-assisted music composition has a history as long as the computer itself. Hiller and Isaacson were the first to use a computer program to compose music in their pioneering work with the Illiac Suite [45]. Over the last six decades, several different techniques and tools have been used to make computers compose and create music [46]. Based on the methods used, these can be roughly categorized into three phases:

(a) **Expert Systems**: The early approaches to computer music composition relied on explicit programming of musical rules to create *expert systems* [45, 47, 48, 49]. The underlying assumption behind such systems was that music composition is essentially an algorithmic process which can be systematically defined by a set of rules [50]. The degree to which these rules were used varied from system to system. For instance, Ebcioglu used an elaborate set of several hundred rules to solve a constraint satisfying problem and generate chorales in the style of Bach [47]. On the other hand, Hiller and Isaacson used rules to only screen the rhythms and pitches generated by a random process to generate their pieces for string quartet [45]. While rule-based systems are generally suitable to achieve specific compositional goals such as counterpoint composition [48], they have certain limitations. Music composition does not always involve following a fixed set of rules. Often, it is the breaking of the rules that result in novel and interesting compositions. In addition, it is not possible to create an exhaustive set of rules for a given style of composition. In fact, musicological rules are only derived by looking backward at the composed pieces.

(b) **Probabilistic Modeling-based Methods**: The next broad category of composition systems considered modeling music composition as a probabilistic or stochastic process. While some composers such as John Cage and Iannis Xenakis took this approach to one extreme through their chance music compositions, a different approach was focused on learning the underlying probability distributions from a collection of musi-

cal data. Given a corpus of music, one can learn the statistical dependence between different musical elements (e.g., the sequence of pitches and rhythms), and then use this to generate new music conforming to the overall musical style of the corpus. This could be accomplished using different methods. For instance, one could use either dictionary-based methods [51, 52], Markov chains [53, 54, 55, 56] or by using neural networks [57, 58, 59].

(c) **Deep Learning-based Methods**: More recently, as a natural extension of probabilistic methods, Deep Neural Networks (DNNs) have emerged as the tool of choice for modeling conditional probabilities from large musical datasets [60, 14, 12, 11, 16]. Given sufficient data to learn from, DNNs can act as powerful function approximators and can potentially learn arbitrarily complex functions. Along with other domains in AI and CC such as text generation, speech generation, image generation, most of the current state-of-the-art music generation systems are exclusively based on deep generative models. Briot et al. provide an overview which demonstrates the rapidly growing popularity of deep learning techniques for music generation over the last decade [10].

1.3 Interactivity and Controllability Challenges

Music creation, in most cases, is essentially a constraint-driven task [61]. Many different aspects of the music creation process rely on some form of constraint being applied such as: *(a)* choice of key/scale, *(b)* choice of instrument, *(c)* dependence of voices and parts on each other, *(d)* dependence on harmony, and *(e)* conforming to a structural plan. The creative choices that a composer/performer makes are influenced by these constraints. Additionally, a typical compositional process involves several iterations where the composer tweaks certain parts/sections based on their aesthetic sensibilities. For instance, the composer might want to change the melody of a particular measure to conform to a different harmony, change the harmony for a particular melodic line to create a different feel, or change the duration

of the notes in a section to create a different rhythm. Thus, for music generation models to be used effectively in practical compositional settings, the users should have the ability to interactively control the generation based on different musical *attributes*.

Musical attributes are certain properties of music. These could be low-level (e.g., note density, pitch range), mid-level (e.g., rhythmic and harmonic complexity, key), or high-level (e.g., musical structure and repetition, emotion, arousal, tension). Attributes could be short-term or long-term, could be objectively computed using the musical score or require subjective human evaluation.

In spite of the progress made by deep generative models in the field of automatic music creation, one of the major criticisms of such models is that, more often than not, they work as black-boxes [18]. The end-users neither have any clear insights into the compositional decisions being made by such models nor have the ability to interact with and control the output of these models in an iterative manner. These music generation systems typically lack in the following areas [18]:

(a) **Interactivity**: In most deep music generative models, the generation process is unidirectional. Neural network-based architectures typically provide a single point of entry to the generation process. Consequently, they do not allow any means to modify a specific part of the generated musical content. Thus, these models offer very few options in terms of interactivity which is a vital aspect of the composition process.

(b) **Control**: The second criticism of deep generative models is that they offer little to no control over the generated content. The internal representations learned by neural networks are often uninterpretable and hence, cannot be effectively used to steer the generated content to explicitly modify a particular musical attribute or meet a specific compositional goal.

Improving interactivity and control, thus, has emerged as a major area of focus for deep music generative models. One approach for improving control has been to add conditioning

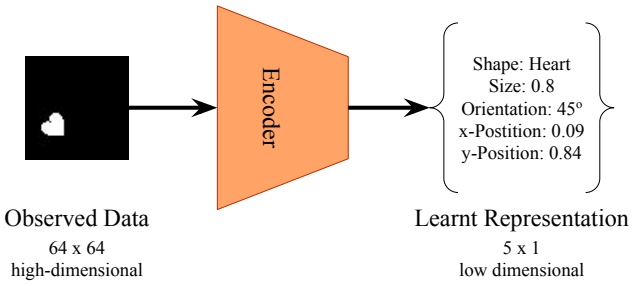


Figure 1.1: Example for Representation Learning. A high dimensional observed data is compressed into a meaningful low dimensional latent representation (sprites image taken from the dSprites dataset [64])

information at the input of the models. For instance, conditioning the generated melody with the underlying chords [14, 62], or conditioning the generated rhythm with the underlying beat structure [63]. Attempts to improve interactivity have relied on allowing users to change part of the input. For instance, the DeepBach model proposed by Hadjeres et al. allows users to not only specify the melody over which to generate 4-part choral harmonies but also selectively regenerate specific voices and parts [60], and the Piano Genie framework proposed by Donahue et al. generates piano scores based on a coarse rhythmic and melodic contour provided by the user [31]. The common thread across the above attempts is to enable conditional generation of music where a user can either impose certain constraints or control different musical attributes so that the generated music conforms to their needs and preferences.

1.4 Motivation for Exploring Latent Representations

The primary motivation for this thesis stems from the need to address the interactivity and controllability challenges facing deep music generative models which were described in the previous section. One of the areas in machine learning research which shows promise in this direction is the field of Representation Learning [19] which deals with learning meaningful and compact representations from data. Figure 1.1 shows a typical example.

In particular, generative models which rely on low-dimensional *latent* representations

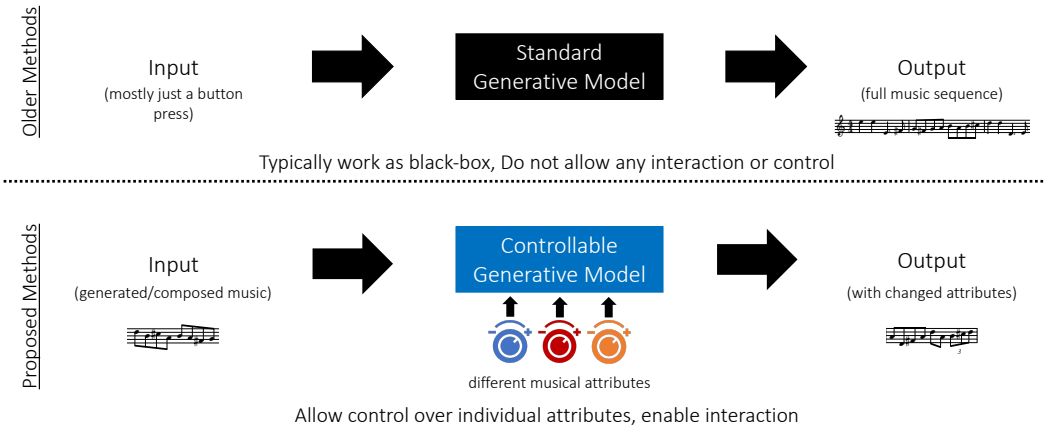


Figure 1.2: Schematic of the overall goal of the proposed research contextualized against older music generation systems.

present some very interesting properties. These models are trained to learn a mapping from a low-dimensional space (referred to as latent space) to high-dimensional data (e.g., the space of all natural images, or the space of all 1-bar melodies) with the expectation that the latent space is able to encode certain attributes of the data. Research across different domains such as computer vision, music, and text generation have shown that learnt latent spaces of generative models may have interesting properties such as:

(a) *Semantic Interpolation*: Semantically meaningful interpolations in the data-space can be achieved by linear interpolations between points in the latent space [11].

(b) *Attribute Vector Arithmetic*: The mean of latent vectors corresponding to data-points sharing a common attribute can be used to add/remove this attribute from a data-point by using simple vector arithmetic operations in the latent space [65, 20].

Indeed, working with a trained latent space, as opposed to raw symbolic data, has also resulted in improvements in several music generation tasks which are reviewed in Section 2.3.

While the properties listed above are useful, which data attributes end up being encoded in these latent spaces is neither clearly understood nor explicitly controlled. Often, latent spaces end up learning *entangled* representations where different attributes of the data are

mixed up along different dimensions of the latent space. From a generative modeling perspective, learning *disentangled* representations is advantageous as it can allow independent control over different attributes during content generation. Therefore, understanding how latent spaces work and being able to manipulate them effectively so as to create interpretable and disentangled latent spaces could be critical for improvements in not only music creation systems but also for other generative modeling applications. Figure 1.2 shows the broad direction of the proposed research.

1.5 Research Questions

The focus of this thesis is to leverage latent representation-based models to address the interactivity and controllability challenges associated with deep generative models of music. Even though several attempts have been made to force semantic structure onto latent spaces in order to improve conditional generation, they suffer from certain limitations which I discuss in Section 2.2. In addition, a vast majority of such methods have been applied to other data domains such as images and their application to musical data remains limited. The specific research questions that drive this work are as follows:

(RQ1) **How can information in latent representations of deep generative models be better used for controlling music generation?**

Latent spaces typically learn entangled representations and it is still unclear if this entangled information present in *vanilla* (basic) latent spaces is enough to learn effective control over different aspects of music generation. There are different methods which can potentially be used to leverage this information better. Instead of using methods which make assumptions regarding the linearity of the latent space (e.g., semantic interpolation and attribute-vector arithmetic), supervised learning methods can be used to learn non-linear trajectories in the latent space. Alternatively, latent spaces can be restructured by using different learning methods so as to disentangle the information available in vanilla latent spaces with respect to specific musical

attributes.

(RQ2) **To what extent can specific musical attributes be encoded in the latent spaces of music generation models?**

While it has been shown that latent spaces can be regularized to encode certain low-level musical attributes (e.g., note density [66, 67]) along specific dimensions, the attributes considered have been typically limited. Some of the methods also impose constraints on how these attributes can be computed. It is not clear if latent representations can be regularized to simultaneously encode multiple attributes along different dimensions. In addition, musical attributes can also have different data-types (e.g., continuous, ordinal, categorical). It is unclear if different methods need to be used for these different data-types or if a single method can work well for everything.

(RQ3) **To what degree can individual musical attributes be disentangled in the latent space?**

So far, disentanglement studies on music data have been unsystematic. They have used different datasets and have been applied to different tasks. Unlike the computer vision domain, there are no standardized datasets to evaluate disentanglement learning methods on music data. Disentanglement of attributes is important from the perspective of conditional music generation. For instance, structuring the latent space so as to give control over a musical attribute (say note density) is only useful when other attributes of the generated music (such as key, scale, underlying melodic motif) remain stable as this attribute is modified. This can be challenging when we consider attributes that are dependent on each other or are strongly correlated.

1.6 Overall Thesis Outline

In this chapter, I described how deep learning-based systems have emerged at the forefront of research in automatic music creation, and how different applications of such systems

have the potential to transform the way music is created. I also discussed the challenges of interactivity and controllability associated with such systems and why overcoming these challenges will be essential to ensure that tools for automatic music creation are able to be used in real-world creative practices. In Section 1.4, I briefly touched upon the potential of latent representation-based models in addressing these challenges and then framed the research questions which are addressed in this thesis in Section 1.5.

Chapter 2 lays the foundations for the rest of this thesis by providing a tailored overview of latent representation-based deep generative models, outlining the different approaches that have been undertaken to improve conditional generation by manipulating latent spaces, and discussing the limitations of such approaches.

The subsequent four chapters deal with the different proposed methods and studies conducted by me which seek to answer the research questions presented in Section 1.5. First, Chapter 3 introduces the *Latent Space Traversal* method. This method involves leveraging the information in existing latent spaces of generative models in a better way to perform the challenging task of connecting two musical excerpts. Next, Chapter 4 presents the *Latent Space Regularization* method which deals with structuring the latent spaces of generative models during the training stage by explicitly encoding specific musical attributes along the individual dimensions of the latent space. This allows such generative models to be used to manipulate musical attributes independently. Subsequently, Chapter 5 presents the *Latent Space Transformation* method which extends the regularization method to existing latent spaces. This is accomplished by learning to transform an existing entangled latent space into a disentangled latent space where different musical attributes can be easily controlled. Next, Chapter 6 deals with the problem of *Disentanglement Learning* in the context of music data by proposing a new symbolic music-based dataset for the task and investigates the performance of several unsupervised and supervised disentanglement learning methods on symbolic music data.

Finally, the core contributions of this thesis are highlighted in Chapter 7 by summarizing

the different results obtained using the proposed methods, and contextualizing the experimental results relative to the research questions. It also presents and discusses avenues for future research and exploration.

The contents of this thesis have been part of the following peer-reviewed publications and demonstrations:

- Ashis Pati, and Alexander Lerch, *"Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders"*, Neural Computing and Applications. 2020. [68]

- Ashis Pati, Siddharth Gururani, and Alexander Lerch, *"dMelodies: A Music Dataset for Disentanglement Learning"*, in Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR). Montréal, Canada. 2020. [69]

- Ashis Pati, Alexander Lerch, and Gaëtan Hadjeres, *"Learning to Traverse Latent Spaces for Musical Score Inpainting"*, in Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR). Delft, The Netherlands. 2019. [70]

- Ashis Pati, and Alexander Lerch, *"Latent Space Regularization for Explicit Control of Musical Attributes"*, in Proceedings of ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract. Long Beach, California, USA. 2019 [71]

- Théis Bazin, Ashis Pati, and Gaëtan Hadjeres, *"A Model-Agnostic Web Interface for Interactive Music Composition by Inpainting"*, Demonstration Track, Neural Information Processing Systems (NeurIPS). Montréal, Canada. 2018. [72]

CHAPTER 2
RELATED WORK

Latent representations are low-dimensional representations learnt from high-dimensional data. As discussed in Chapter 1, the overall goal of this thesis is to address the interactivity and controllability challenges associated with deep music generative models by learning to manipulate these latent representations.

This chapter provides a tailored review of current literature in leveraging latent spaces for different generative tasks. It begins with a formal introduction to deep generative models in Section 2.1. This is followed by a description covering the basics of common latent representation-based deep generative models such as Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs). The limitations of the latent representations learnt by the vanilla versions of these frameworks are discussed. Next, Section 2.2 provides an overview of the different research directions undertaken in the machine learning community to make latent spaces more interpretable with respect to the different data attributes. Specifically, both unsupervised and supervised techniques are covered and their respective strengths and weaknesses are discussed. Subsequently, Section 2.3 briefly describes the benefits of using latent spaces in the context of music generation and MIR tasks. It also discusses how some of the methods described previously in Section 2.2 have been successfully applied to music generation tasks. Finally, Section 2.4 presents a description of the different disentanglement metrics which are used in the experiments conducted in this thesis.

2.1 Background on Deep Generative Models

Generative modeling is an area of machine learning which deals with learning probability distributions over high dimensional data [73], e.g., the space of all natural images or the space of all 1-bar melodies. Formally, given data-points $\mathbf{x} \in X$, where X is high-dimensional, the

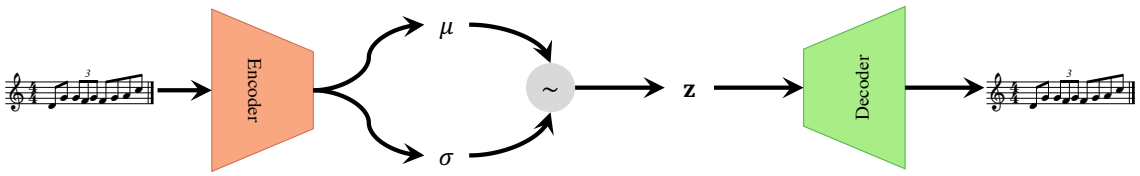


Figure 2.1: Schematic of a typical VAE architecture. Encoder and Decoder are neural networks. The Encoder outputs the mean μ and standard deviation σ which parametrize the latent distribution $p(\mathbf{z})$. The latent vector $\mathbf{z}$ is obtained by the sampling operation $\sim$.

task is to estimate the likelihood $p(\mathbf{x})$. Simply put, the core idea is that if the probability distributions over the data-space can be learned or estimated, then new data could be generated by sampling from these distributions.

Deep generative models leverage the representational power of neural networks to approximate these distributions from data. Over the last decade, several different techniques have been developed which include auto-regressive networks [74], VAEs [75], GANs [76], and flow-based models [77]. Developments in deep generative models have led to improvements in a variety of tasks ranging from image [78] to text [79] to music generation [11]. In addition, generative models have also been used for other tasks such as semi-supervised learning [80, 81], and representation learning [82, 83].

VAEs and GANs in particular use a latent representation, i.e., a low dimensional space from which points can be sampled to map to the high dimensional data-space. A brief description of VAE and GAN frameworks is presented below.

2.1.1 Variational Auto-Encoder (VAE)

A VAE [75] is a type of generative model which uses an auto-encoding [84] framework — during training, the model is forced to reconstruct its input. In a typical auto-encoder, the encoder learns to map data-points $\mathbf{x}$ from a high-dimensional data-space X to points in a low-dimensional space Z. This low-dimensional space is referred to as the *latent* space, and points $\mathbf{z}$ in the latent space are called *latent vectors* (or latent embeddings).

The decoder learns to map the latent vectors back to the data-space. VAEs treat the latent vector as a random variable and model the generative process as a sequence of sampling operations: $\mathbf{z} \sim p(\mathbf{z})$, and $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$, where $p_\theta(\mathbf{x}|\mathbf{z})$ is the decoder parametrized by θ, and $p(\mathbf{z})$ is a prior distribution over the latent space (see Figure 2.1). The posterior $p(\mathbf{z}|\mathbf{x})$ is intractable. Variational inference is then used, which consists of introducing $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the posterior with an encoder parameterized by ϕ. Both the encoder and decoder are implemented using neural networks. The approximation is done by minimizing the Kullback-Leibler (KL)-divergence [85] between the approximate posterior and the true posterior by maximizing the evidence lower bound (ELBO).

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.1}$$

where $\mathbb{E}[\cdot]$ is the mathematical expectation, $D_{\mathrm{KL}}(\cdot||\cdot)$ is the KL-divergence.

The first term of Equation 2.1 can be interpreted as maximizing the reconstruction accuracy while the second term ensures that realistic samples are generated when latent vectors are sampled using the prior $p(\mathbf{z})$ [11]. In practice, the VAE training process minimizes the following loss function:

$$L_{\mathrm{VAE}}(\theta, \phi) = L_{\mathrm{recons}}(\theta, \phi) + L_{\mathrm{KLD}}(\phi) \tag{2.2}$$

where $L_{\mathrm{recons}}(\theta, \phi)$ and $L_{\mathrm{KLD}}(\theta, \phi)$ are the VAE reconstruction loss and the KL-Divergence regularization, respectively.

The reconstruction loss is defined as:

$$L_{\mathrm{recons}}(\theta, \phi) = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \tag{2.3}$$

where N is the number of examples, $\hat{\mathbf{x}}$ is the reconstruction of $\mathbf{x}$ obtained using the encoder and decoder of the VAE. The L^2-norm in the above equation can be replaced by cross-entropy

loss when the reconstruction is over a categorical distribution.

The regularization loss is given by:

$$L_{\mathrm{KLD}}(\phi) = D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \tag{2.4}$$

The auto-encoding style training process of a VAE, where a high-dimensional data-point is compressed into a low-dimensional latent space, forces the network to extract meaningful attributes from the data. A limitation of this training process, however, is that the model arbitrarily decides which attributes to encode along which dimensions of the latent space. Consequently, the learnt latent spaces are often entangled. This lack of interpretability makes a vanilla-VAE less useful from a conditional generation perspective where specific attributes of the generated data need to be modified.

One of the more common variants of the VAE model which has been used to force disentanglement is the β-VAE model [86, 87]. It uses the following formulation:

$$L_{\mathrm{VAE}}(\theta, \phi) = L_{\mathrm{recons}}(\theta, \phi) + \beta L_{\mathrm{KLD}}(\phi). \tag{2.5}$$

The core idea here is that using $\beta > 1$ encourages the independence of the dimensions of the latent space and leads to better disentanglement. However, the trade-off is that increasing β might result in reduced reconstruction quality.

2.1.2 Generative Adversarial Networks (GAN)

GANs [76] rely on using the discriminative power of neural networks to generate new data-points which match the distribution of real data. They achieve this without explicitly learning the underlying probability distributions. The framework has a Generator G which maps a randomly sampled low-dimensional noise $\mathbf{z}$ to the so-called fake samples $\mathbf{x}_f$ and a Discriminator D has to discriminate between the fake samples and the real samples $\mathbf{x}_r$ taken from actual data (see Figure 2.2). During the GAN training process, as G and D

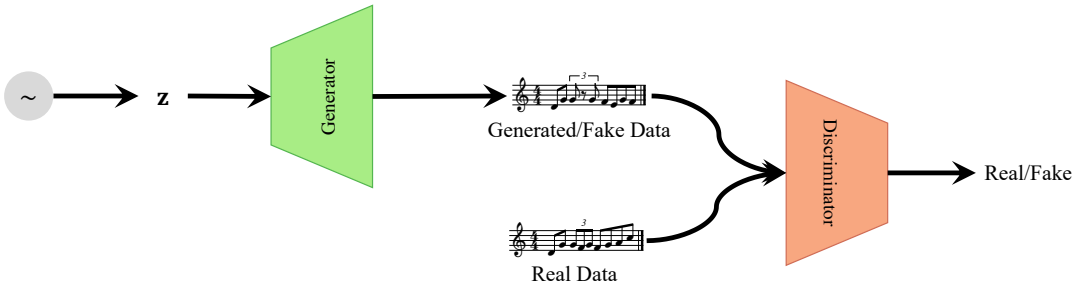


Figure 2.2: Schematic of a typical GAN architecture. Generator and Discriminator are neural networks. The Generator uses the latent vector $\mathbf{z}$ produced from a sampling operation $\sim$ to create fake data-points and learns to fool the Discriminator which in turn learns to discriminate between the real and fake data-points.

engage in a mini-max game, D gets better at discriminating the fake samples from the real samples whereas G gets better at fooling D by producing realistic samples. Both G and D are implemented using neural networks.

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.6}$$

GANs, like VAEs, operate on a low dimensional latent space Z from which the noise $\mathbf{z}$ is sampled. The key difference is that, unlike VAEs, there is no way to estimate the posterior distribution $p(\mathbf{z}|\mathbf{x})$. This means that, given a data-point $\mathbf{x}$, it is not possible to estimate the latent vector $\mathbf{z}$ which has the maximum likelihood to generate $\mathbf{x}$. The GAN framework in its most basic form also learns entangled representations to map the latent vector $\mathbf{z}$ to the data-space [88]. Hence, like VAEs, they lack interpretability and thus, cannot be used for conditional generation directly.

2.2 Improving Interpretability of Latent Representations

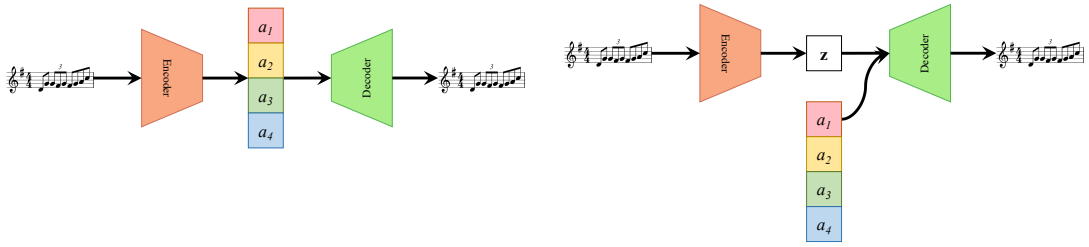
Improving the interpretability of latent representations has been an active area of research in the deep learning community and several methods have been developed to achieve this objective. These can be roughly categorized into three broad groups. The first group of methods attempts to disentangle different factors of variation in the data [19]. The majority

of the methods in this category are unsupervised techniques. Approaches from the second category rely on identifying certain attributes of interest and using supervised techniques in order to enable control during the generation process. The third category deals with working on top of existing (possibly entangled) latent spaces and use different methods to improve interpretability. These can either be transformation or traversal-based methods.

2.2.1 Unsupervised Disentanglement Learning

Unsupervised methods for disentanglement learning attempt to separate the distinct factors of variation in data [19] and learn a representation where changes to a single underlying factor of variation (attribute) lead to changes in a single factor (dimension) of the learned representation [89]. All this is done without relying on any extra information about the attributes. Most of the current approaches to unsupervised disentanglement are based on variations of the VAE framework [75]. The main idea behind these approaches is that forcing the latent representation to have a factorized aggregated posterior should result in disentanglement [89]. This can be achieved using different means such as imposing constraints on the information capacity of the latent space [86, 90, 91], maximizing the mutual information between a subset of the latent code and the observations [82], and maximizing the independence between the latent variables [92, 93].

While many of these methods show good performance (based on one or more objective metrics for measuring disentanglement) on artificially generated datasets (such as dSprites [64]), a recent study by Locatello et al. shows that not only are these approaches sensitive to inductive biases such as choice of network, hyperparameters, and random seeds but also that some amount of supervision is necessary for learning effective disentanglement [89]. In addition, since these methods seek to learn a factorized latent representation they work well for low-level data attributes. However, as shown in the experiments in this thesis, they do not extend well for complex data attributes (which are usually some combination of low-level attributes). Consequently, their practical usefulness is limited. Using unsupervised methods



(a) Regularization-based methods (b) Conditioning-based methods

Figure 2.3: Schematic showing difference between regularization and conditioning based methods for a VAE framework. a_i denotes the i^{th} attribute. For regularization-based methods (left), the latent space is decomposed into parts which encode different attributes. For conditioning-based methods (right), attribute values are provided as conditional inputs during the generation process.

for manipulating attributes also requires a post-training analysis to determine how different attributes are encoded along the different dimensions of the latent space.

2.2.2 Supervised Learning Methods

This set of methods relies on using extra information about the different attributes of the data. There has been some research on supervised methods to control attributes by directly learning the exact data transformation needed to change a given attribute by a particular amount [94, 95]. However, these methods require huge amounts of finely annotated data mapping each transformation. Obtaining such data is costly and hence, such methods are of limited practical use.

An alternative is to use any available attribute information for supervised training in an implicit manner. Methods within this group can broadly be differentiated into two categories (also shown in Figure 2.3):

(a) **Regularization-based methods:** In these methods, individual (or a sub-set of) dimensions of the latent space are regularized to encode different attributes. One example is the InfoGAN architecture which uses implicit encoding [82]. Specifically, it maximizes the mutual information between certain dimensions of the latent vector and

the generated data-points. This forces the network to use those dimensions to only modify certain intrinsic attributes of the data-points. InfoGANs trained on images have been found to encode attributes such as rotation, lighting, etc. The limitation of this approach, however, is that there is no way to select which attributes end up being encoded. An alternative to this is to use explicit regularization techniques such as the Geodesic Latent Space Regularization (GLSR) proposed by Hadjeres et al. [67]. This method imposes a regularization loss function to encode a selected attribute along a specific dimension of the latent space. However, the loss formulation requires differentiable computation of the attributes and extensive hyperparameter tuning. Donahue et al. proposed a method to encode the facial identity of a person [96] in the latent space of a GAN-based model trained to generate facial images. Instead of using just one dimension, they decomposed the latent space into two parts: one encoded variation in facial identity, and the second encoded variation due to all other attributes.

(b) **Conditioning-based methods:** These methods use additional attribute-specific conditioning information during the training and generation process. The idea here is to learn a generalized latent representation which, when combined with the attribute-specific conditioning information, creates a data-point with that attribute. Conditional VAEs [97, 98] and Conditional GANs [99] were early attempts at this. The more recent Fader networks proposed by Lample et al. demonstrated the effectiveness of this idea for image generation by using an adversarial training scheme [100]. This approach allows free modification of several attributes during inference time.

Supervised methods also have their limitations. On the one hand, some methods are constrained to work only with certain types of data attributes. For instance, the Fader network, although designed for categorical attributes, has been shown to only work well with binary attributes [100]. On the other hand, some methods might impose additional constraints. For example, they might require the ability to generate data-points by independently varying

attributes [83], require differentiable computation of attributes [67], or require the ability to group data-points based on certain attributes [101, 96]. In addition, few supervised approaches are designed to work with continuous-valued attributes [67, 102].

2.2.3 Transformation and Traversal-based Methods

This category of methods work on top of existing (possibly entangled) latent spaces and use different methods to improve their interpretability.

One approach for this is to find an *attribute vector* which encodes a high-level feature or concept and then add/subtract this from a latent vector to result in samples with/without this feature [103, 104, 11]. A criticism of this approach is that it makes a simplistic assumption that simple vector arithmetic on the latent space would be sufficient to model complex semantic attributes [67].

Subsequently, researchers have adopted techniques to learn possible non-linear transformations between the attributes, the latent spaces, and the generated data using neural networks. In one such approach, Engel et al. trained a GAN-style generator-discriminator framework on the latent space of a trained VAE [66]. Conditional generation was enforced by using conditioning inputs similar to the conditional-GAN framework. Adel et al. proposed the creation of a *Lens* model which can map the learnt latent space of a VAE model to a secondary latent space [105]. This secondary latent space, conditioned on attributes of interest, can be leveraged to control the generation during inference time.

The major benefit of transformation-based methods is that they do not interfere in the training process of the existing latent space. Consequently, the reconstruction fidelity is not sacrificed. In addition, the existing models do not need to be retrained when new attributes are added. They can also be potentially used in cases where the attribute information is available for only part of the dataset. A downside of these methods is that they have to start from a compressed entangled representation where all the necessary information might not be available.

2.3 Latent Spaces of Music-based Models

Latent representation-based models have been found to be quite useful for several music generation and MIR tasks. Bretan et al. used the latent representation of an auto-encoder-based model trained to reconstruct a set of musical features and coupled it with a unit selection methodology to generate musical phrases [106]. Lattner et al. forced the latent space of a gated auto-encoder to learn pitch interval-based representations which improved the performance of predictive models of music [107] and was also applied successfully to audio-to-score alignment tasks [108]. Bretan and Heck trained a latent representation-based model to force semantically similar (neighboring) musical phrases to be closer in the latent space [109]. The learnt latent representation was shown to perform well across a range of downstream tasks such as composer classification, pitch chroma prediction, and note density regression.

In the context of music generation, the MusicVAE model by Roberts et al. was one of the first to showcase attribute vector arithmetic in a latent space trained on monophonic symbolic music [11] to control low-level attributes such as note density. Subsequently, Simon et al. proposed an extension of this model for multi-track music and were able to generate music for one instrument conditioned on the others [110]. The MidiNet architecture proposed by Yang et al. was the first to use a GAN-based framework to generate music [14]. Along with the input noise vector, they used conditioning inputs corresponding to the chords as well as information from the previous bar to generate music conforming to these constraints.

Some of the techniques discussed in Section 2.2 for improving the interpretability of latent spaces have also been applied to music generation models. The GLSR technique proposed by Hadjeres et al. allowed finer control over a couple of low-level musical attributes — note-density and pitch-range [67]. Brunner et al. showed that the latent space of their MIDI-VAE model could be regularized to encode genre-specific information along

a sub-set of latent dimensions [111]. They used this technique to perform style transfer between a couple of genres. Yang et al. used a similar idea to force the latent space of a VAE-based model to separately encode pitch and rhythmic information [112]. This was later used to perform rhythm transfer from one monophonic melody to another. The GAN-based technique proposed by Engel et al. used specific reward functions to constrain the generation of their VAE model in two ways. The model could be used to generate notes conforming to a particular scale, or have a certain density of notes.

The success of these methods in music generation tasks further motivates the need for a deeper exploration of latent representations of music in this thesis.

2.4 Disentanglement Metrics

One of the major focus areas of this thesis is on understanding disentanglement in the context of music generation models (see RQ3). There has been considerable work within the machine learning community in the recent past to define objective metrics for measuring the disentanglement of latent spaces [105, 113, 92, 114, 86, 93, 115]. The different metrics which are used across experiments in this thesis are described in the sub-sections below.

To ensure consistency the following terminology is used for the rest of this section. Given a data-point $\mathbf{x}$, a learnt representation $\mathbf{z} = r(\mathbf{x})$ maps $\mathbf{x}$ to a $\mathbb{D}$-dimensional latent space where $\mathbf{z} : \left\{z^k\right\}, k \in [0, \mathbb{D})$. Here $r(\cdot)$ is function mapping $\mathbf{x}$ to $\mathbf{z}$, and is typically implemented using neural networks. Each data-point can be described by a ground-truth factor (attribute) set $\mathcal{A}$ which consists of $\mathbb{L}$ attributes $\mathcal{A} : \{a_l\}, l \in [0, \mathbb{L})$.

2.4.1 Interpretability

Adel et al. consider a latent space to be interpretable with respect to an attribute if a simple linear relationship can be used to explain this attribute [105]. For a given attribute a_l, the interpretability metric M_l is computed using the following two-step process:

(a) First, the dimension of the latent space i which is maximally informative about the

attribute is determined: $i = \text{argmax}_k I(a_l, z^k)$, where $I(\cdot)$ is the mutual information.

(b) The interpretability metric is then computed as $\text{M}_l = p(a_l|z^i)$, where p is a simple linear relationship. For continuous attributes, this is computed using linear regression. For discrete attributes, a simple linear classifier is used.

The final metric is computed by averaging across all attributes.

2.4.2 Mutual Information Gap (MIG)

This metric proposed by Chen et al. assumes that a latent space is disentangled with respect to an attribute a_l if there exists a single dimension which is maximally informative about this attribute [92]. This is computed by using the average, normalized difference of the highest and second-highest mutual information of a_l with all the dimensions in the latent space:

$$\text{MIG} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{H_{a_l}}(I(z^{i_m}, a_l) - \max_{i \neq i_m} I(z^i, a_l)) \tag{2.7}$$

where, $i_m = \text{argmax}_i I(z^i, a_l)$, H_{a_l} is a normalizing factor obtained by summing the mutual information of a_l over all the dimensions of the latent space.

2.4.3 Modularity

Ridgeway and Mozer define a modular representation as one where each dimension of the representation $\mathbf{z}$ depends at most on a single attribute [113]. *Modularity* is defined as the mean of the normalized squared difference of mutual information between the top two attributes which have maximum mutual information with a latent dimension. This is in essence the exact opposite of MIG which tries to measure the degree to which an attribute depends on a single dimension of the latent space.

First, a matrix $\mathbf{m} \in \mathbb{R}^{\mathbb{D}\times\mathbb{L}}$ is computed which stores the mutual information between each attribute and each dimension of the latent space. For each dimension of the latent space

k, a vector $\mathbf{t}_k$ is computed as:

$$t_{k,l} = \begin{cases} \theta_k & \text{if } l = \text{argmax}_g m_{k,g} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where, $\theta_k = \max_g m_{k,g}$.

The *Modularity* score is defined as: $\sum_{l=1}^{\mathbb{L}}(1-\delta_k)$, where $\delta_k = \frac{\sum_l (m_{k,l}-t_{k,l})^2}{\theta_i^2(\mathbb{L}-1)}$.

2.4.4 Separated Attribute Predictability (SAP)

The *SAP* score, introduced by Kumar et al. evaluates the degree to which the value of a given attribute a_l can be predicted by a single dimension of the latent space [114]. This is done by computing the difference in the predictive power of the top two most predictive dimensions of latent space. For continuous attributes, the predictive power is measured using the R^2 score of a linear regression model. For discrete attributes, a trained classifier is used. The final metric is computed by averaging over the different attributes:

$$\textit{SAP} = \frac{1}{\mathbb{L}} \sum_{l=1}^{\mathbb{L}} (p(a_l|z^{i_m}) - \max_{i \neq i_m} p(a_l|z^i)) \tag{2.9}$$

where, $i_m = \text{argmax}_i p(a_l|z^i)$, and $p(y|x)$ represents either the R^2 score or the accuracy of the classifier. SAP is very similar to MIG except while the former uses classifiers (or linear regression) to measure performance, the latter uses mutual information.

2.4.5 Spearman Correlation Coefficient

This metric is exclusively used for continuous attributes to evaluate the degree of monotonicity between the attributes and the latent dimensions. For a given attribute a_l, this is measured by computing the maximum value of Spearman's correlation coefficient between the attribute and all the dimensions of the latent space. The final metric is obtained by averaging across all attributes.

$\textit{SCC} = \frac{1}{\mathbb{L}} \sum_{l=1}^{\mathbb{L}} \max_k \rho(a_l, z_k)$

where, $\rho(\cdot)$ represents the Spearman correlation coefficient.

2.4.6 Implementation Details

For the experiments in this thesis, the above metrics are computed using data-points from held-out test sets. For *MIG*, *Modularity*, and *SAP*, standard implementations used by Locatello et al. [89] are adopted.

CHAPTER 3
LATENT SPACE TRAVERSAL

Many of the deep learning-based music generation models typically assume sequential generation of music, i.e, the generated music depends only on the music that has preceded it. In other words, the models rely only on the past musical context. This approach does not align with typical human compositional practices which are often iterative and non-sequential in nature. In addition, the sequential generation paradigm places severe limitations on the degree of interactivity allowed by these models [116, 10]. Once generated, there is no way to tweak specific parts of the generation so as to conform to the users' aesthetic sensibilities or compositional requirements.

This problem can be addressed by incorporating future musical context into the generation process. One way to do this would be to train models to fill in missing information in musical scores, duly taking into account the complete musical context — both past and future. This task is referred to as *inpainting*, where the objective is to reconstruct missing or degraded parts of any kind of media [117]. For music, inpainting has been traditionally used for restoration purposes [118] or to remove unwanted artifacts such as clipping [119, 120] and packet loss [121]. However, models for *Musical Score Inpainting* (see Figure 3.1) can be used as tools for music creation which can aid people in *(a)* getting new musical ideas based on specific styles, *(b)* joining different musical sections together, and *(c)* modifying or extending solos. In addition, such models can allow interactive music generation by enabling users to change the musical context and get new suggestions based on the updated musical information.

As discussed in Chapter 2, latent representation-based deep generative models are trained to learn low-dimensional representations from high-dimensional data and can encode specific attributes of the data [20]. This chapter presents a novel method which relies on

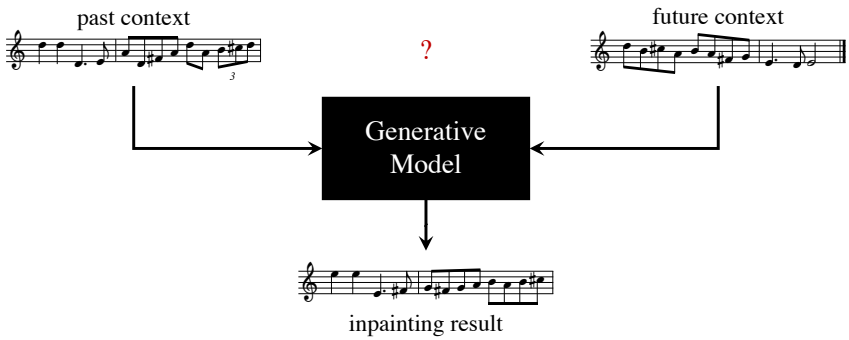


Figure 3.1: Schematic of the Musical Score Inpainting task. A generative model needs to take past and future musical contexts into account to generate a sequence that can connect them in a musically meaningful manner.

leveraging the encoded information in the latent spaces of VAE-based generative models to address the musical score inpainting task. The proposed model uses Recurrent Neural Networks (RNNs) trained on latent vectors to learn complex trajectories in the latent space. This, in turn, is used to predict how to fill in missing measures in a piece of symbolic music. The effectiveness of the proposed method is demonstrated using several objective and subjective evaluation methods.

The remainder of this chapter[1] is organized as follows: first, a brief overview of the related work is presented in Section 3.1 followed by the formal definition of the musical score inpainting problem in Section 3.2. Subsequently, the proposed method covering the model architectures, data representation, and training scheme is covered in Section 3.3. The details of the experimental set-up and the results are provided in Section 3.4 and Section 3.5, respectively. Finally, Section 3.6 summarizes the results.

3.1 Related Work

The first applications of audio inpainting methods were restoration-oriented [118, 119, 122, 121, 123], using different methods such as matrix factorization [119], non-local similarity measures [123], and audio similarity graphs [121]. While these techniques have been useful

[1] Parts of this chapter have been published in [70] and [72].

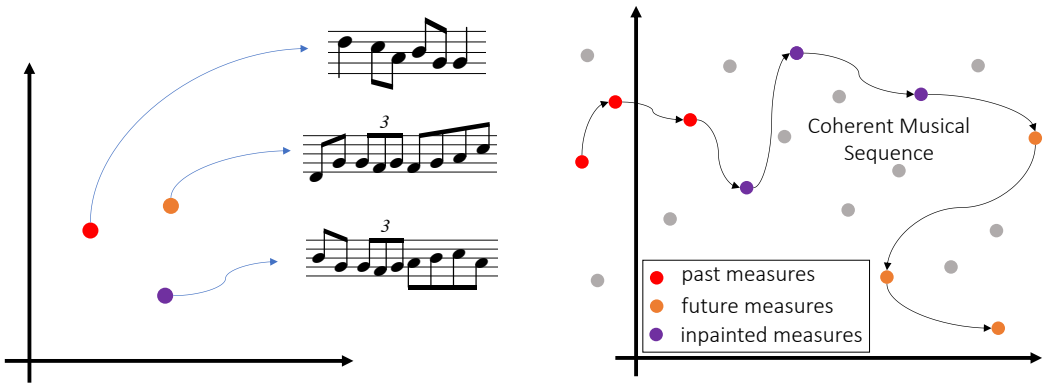
for audio-based tasks, they are not easily extendable to symbolic music.

For inpainting in the symbolic domain, the early attempts were based on Markov Chain Monte Carlo (MCMC) methods which allowed users to specify certain constraints, e.g., which notes to generate and which to retain [55, 124, 60]. Another approach, proposed by Lattner et al., uses iterative gradient descent to force the output of a deep generative model to conform to a specified structural plan [125]. However, methods based on MCMC (which rely on repeated sampling), and those using iterative gradient descent are slow during inference time and hence unsuitable for interactive applications. The AnticipationRNN framework [116] uses a pair of stacked RNNs to enforce user-defined constraints during inference. This allows selective regeneration of specific parts of the music (generated or otherwise) using only two forward passes through the RNN-pair and enabled real-time generations.

Latent representation-based models such as VAEs have been found to be quite useful for several music generation tasks. However, methods based on latent space traversals have relied on simpler approaches such as attribute vectors [65, 20] or linear interpolations [126]. Both methods use linear trajectories in the latent space. If a musical sequence is to be generated by traversing through the latent space, then using linear trajectories makes it impossible to model musical repetitions. This makes attribute vectors and linear interpolations unsuitable for the task of music inpainting.

3.2 Problem Statement

The score inpainting problem, shown earlier in Figure 3.1, is formally defined as follows: given a past musical context $\mathcal{C}_{\mathrm{p}}$ and a future musical context $\mathcal{C}_{\mathrm{f}}$, the modeling task is to generate an inpainted sequence $\mathcal{C}_{\mathrm{i}}$ which can connect $\mathcal{C}_{\mathrm{p}}$ and $\mathcal{C}_{\mathrm{f}}$ in a musically meaningful manner. In other words, the model should be trained to maximize the likelihood $p(\mathcal{C}_{\mathrm{i}} \mid \mathcal{C}_{\mathrm{p}}, \mathcal{C}_{\mathrm{f}})$. Without much loss of generality, it is assumed that $\mathcal{C}_{\mathrm{p}}$, $\mathcal{C}_{\mathrm{f}}$, and $\mathcal{C}_{\mathrm{i}}$ comprise of n_{p}, n_{f}, and n_{i} measures of music, respectively.



(a) Measure-VAE Latent Space (b) Traversing the MeasureVAE Latent Space

Figure 3.2: Illustration of the proposed Latent Space Traversal approach for music inpainting. Each point in the MeasureVAE latent space (shown on the left) encodes a single measure of music. The LatentRNN model learns to find complex trajectories in this latent space to connect the past and future contexts in a musically meaningful manner.

3.3 Method

The key assumption behind the proposed method is that the latent representations learnt by deep generative models of music can end up encoding hidden attributes of music which can be leveraged to perform inpainting. Firstly, a VAE-model, referred to as *MeasureVAE*, is trained to reconstruct single measures of music, i.e., the latent vectors of this model $\mathbf{z} \in Z$ map to individual measures of music (see left of Figure 3.2). Once trained, the encoder of this model can be used to process sequences $\mathcal{C}_{\mathrm{p}}$ and $\mathcal{C}_{\mathrm{f}}$ and output corresponding latent vector sequences $\mathcal{Z}_{\mathrm{p}}$ and $\mathcal{Z}_{\mathrm{f}}$. Secondly, an RNN-based model, referred to as *LatentRNN*, is trained to take as input the past and future latent vector sequences ($\mathcal{Z}_p$ and $\mathcal{Z}_{\mathrm{f}}$) and output a third latent vector sequence $\mathcal{Z}_{\mathrm{i}}$ which can be passed through the decoder of MeasureVAE to obtain $\mathcal{C}_{\mathrm{i}}$.

Effectively, the LatentRNN model learns to traverse the latent space of the MeasureVAE model so as to connect the provided contexts in a musically meaningful manner (see right of Figure 3.2). The inference is fast since it only requires forward passes through the two models. The overall schematic of the joint architecture, referred to as *InpaintNet*, is shown

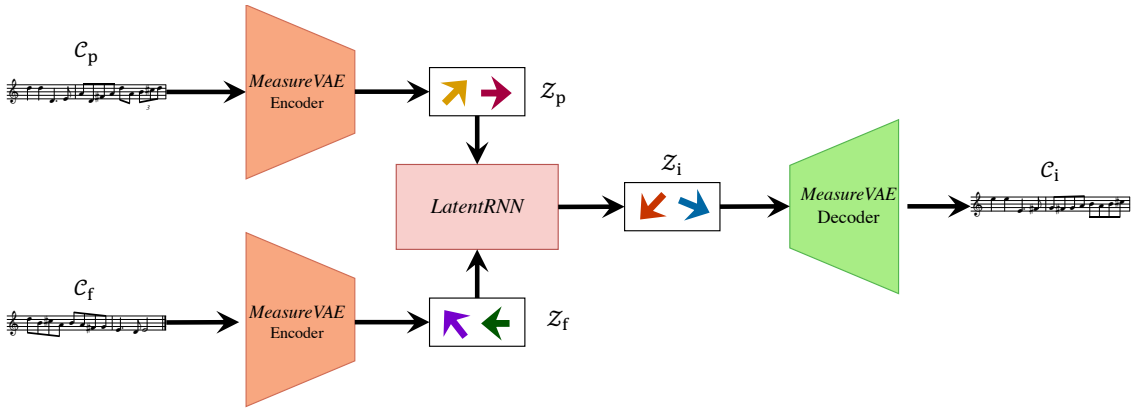


Figure 3.3: Schematic of the proposed model architecture for Inpainting. The pre-trained MeasureVAE encoder is used to convert the past and future context sequences ($\mathcal{C}_\mathrm{p}$ and $\mathcal{C}_\mathrm{f}$) into their respective latent vector sequences ($\mathcal{Z}_\mathrm{p}$ and $\mathcal{Z}_\mathrm{f}$). The LatentRNN learns to traverse the latent space of MeasureVAE to output a latent vector sequence $\mathcal{Z}_\mathrm{i}$ which is passed through the pre-trained decoder to output the inpainted musical sequence $\mathcal{C}_\mathrm{i}$.

in Figure 3.3. Even though only $4/4$ monophonic melodic sequences are considered in this work, the approach can be extended to other time signatures and polyphonic sequences as well. The details of the individual model architectures are discussed next.

3.3.1 Model Architectures

MeasureVAE: The MeasureVAE architecture (see figure 3.4) is loosely based on the hierarchical recurrent MusicVAE architecture [11] which proved successful in modeling individual measures of music.

The input to the encoder consists of a sequence of musical tokens. The encoder consists of a learnable embedding layer (operating at the token level) followed by a bi-directional RNN [127]. The concatenated hidden state from both directions of the RNN is then passed through two identical parallel linear stacks to obtain the mean μ and variance σ^2 which are used to sample the latent vector $\mathbf{z}$ via $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, where $\mathcal{N}(\cdot)$ denotes the Gaussian distribution.

The decoder follows a hierarchical structure where the sampled latent vector $\mathbf{z}$ is used to initialize the hidden state of a beat-RNN which is unrolled b times (where b is the number

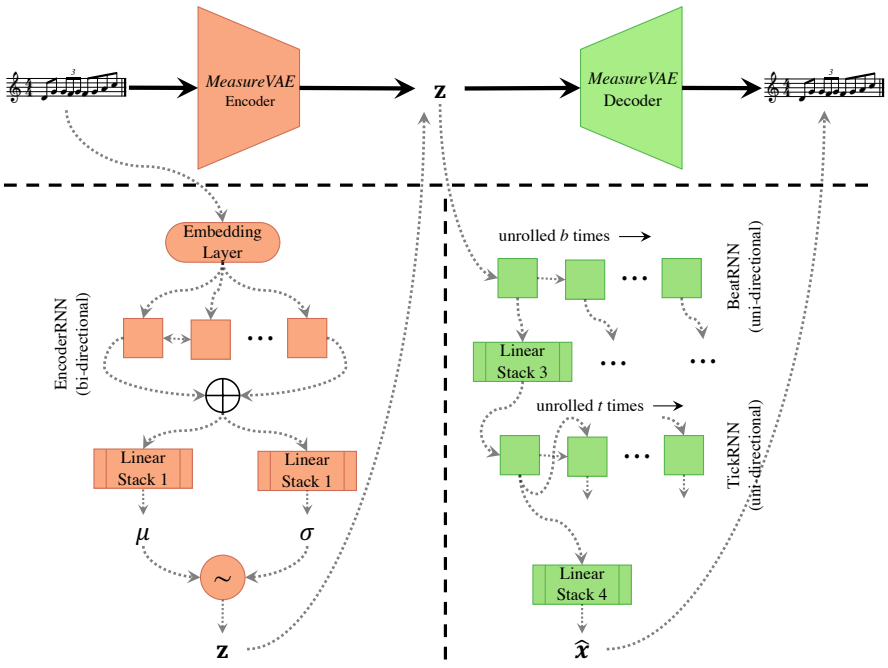


Figure 3.4: Schematic of the MeasureVAE architecture. Individual components of the encoder and decoder are shown below the main blocks (dotted arrows indicate data flow within the individual components). $\mathbf{z}$ denotes the latent vector and $\hat{\mathbf{x}}$ denotes the reconstructed measure.

of beats in a measure). The output at each step of the beat-RNN is passed through a linear stack before being used to initialize the hidden state of a tick-RNN which is unrolled t times (where t is the number of events/ticks in a beat). The outputs of the tick-RNN are individually passed through a second linear stack which maps them back to the data-space. As advocated by Roberts et al. [11], the hierarchical architecture mitigates the auto-regressive nature of the RNN and forces the decoder to use the latent vector more efficiently.

LatentRNN: The LatentRNN model (see figure 3.5) consists of 3 sub-components. There are 2 identical bi-directional RNNs, referred to as Past-Context-RNN and Future-Context-RNN, which process the latent vector sequences for the past and future contexts ($\mathcal{Z}_{\mathrm{p}}$ and $\mathcal{Z}_{\mathrm{f}}$), respectively. These are unrolled for n_{p} and n_{f} times, respectively, in order to encode the context sequences. The final hidden states of the two context RNNs are concatenated and then used to initialize the hidden state of a third RNN, referred to as the Generation-RNN, which is unrolled n_{i} times. The outputs of the Generation-RNN are passed through a linear

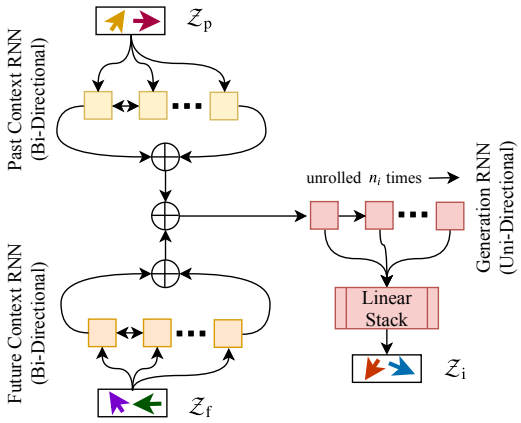


Figure 3.5: Schematic of the LatentRNN architecture. The Past-Context and Future-Context-RNNs encode $\mathcal{Z}_{\mathrm{p}}$ and $\mathcal{Z}_{\mathrm{f}}$, respectively. The Generation-RNN initialized using a concatenation of context-RNNs embeddings is unrolled n_{i} times to get $\mathcal{Z}_{\mathrm{i}}$.

stack to obtain n_{i} latent vectors corresponding to the inpainted measures.

The hyper-parameters for the model configurations are chosen based on initial experiments and are provided in Table 3.1. For the RNN layers in both models, Gated Recurrent Units (GRU) [128] are used.

3.3.2 Stochastic Training Scheme

A stochastic training scheme is used for training the model. For each training batch, the number of measures to be inpainted (n_{i}) and the number of measures in the past context (n_{p}) are randomly sampled from a uniform distribution. Thus, the number of measures in the future context becomes $n_{\mathrm{f}} = N - n_{\mathrm{i}} - n_{\mathrm{p}}$, where N is the total number of measures in each sequence of the training batch. Using these, the input sequences are split into past, future, and target sequences. The model is trained to predict the target sequence given the past and future context sequences. This stochastic training scheme ensures that the model learns to deal with variable length contexts and can perform inpaintings at arbitrary locations.

Table 3.1: Configurations of MeasureVAE and LatentRNN models. n: Number of Layers, i: Input Size, o: Output Size, h: Hidden Size, d: Dropout Probability, SELU: Scaled Exponential Linear Unit [129], ReLU: Rectifier Linear Unit

Measure VAE	
Embedding Layer	i=dict size, o=10
EncoderRNN	n=2, i=10, h=512, d=0.5
Linear Stack 1 Linear Stack 2	i=1024, o=256, n=2, non-linearity=SELU
BeatRNN	n=2, i=1, h=512, d=0.5
TickRNN	n=2, i=522, h=512, d=0.5
Linear Stack 3	i=512, o=1024, n=1, non-linearity=ReLU
Linear Stack 4	i=512, o=dict size, n=1, non-linearity=ReLU
Latent RNN	
Past-Context-RNN Future-Context-RNN	n=2, i=256, h=512, d=0.5
Generation RNN	n=2, i=1, h=1024, d=0.5
Linear Stack	i=2048, o=256, n=1, non-linearity=None

3.3.3 Data Encoding Scheme

A variant of the encoding scheme proposed by Hadjeres et al. [60] is used to represent the music data. The original encoding scheme quantizes time uniformly using the sixteenth note as the smallest sub-division. For each sub-division or tick, the note which starts on that tick is represented by a token corresponding to the note name. If no note starts on a tick, a special continuation symbol '__' is used to denote that the previous note is held. Rest has a special token. This encoding scheme uses a single sequence of tokens and uses real note names (e.g., separate tokens for A# and Bb) which allows generation of readable sheet music.

However, a limitation of using the sixteenth note as the smallest sub-division is that it cannot encode triplets. The naive approach of evenly subdividing the sixteenth note divisions to encode triplets increases the sequence length a factor of 3 which can make the sequence modeling task harder. To mitigate this limitation, an uneven subdivision scheme is used.

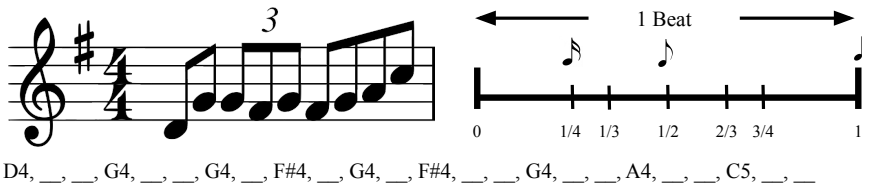


Figure 3.6: Data representation scheme for the inpainting task. The token string on the bottom demonstrates the encoding scheme for the measure displayed on the left. Right shows the proposed uneven tick-duration scheme for each beat.

Each beat is divided into 6 uneven ticks (shown in Figure 3.6). This allows encoding triplets while only increasing the sequence length by a factor of 1.5. Consequently, each $4/4$ time signature measure is a sequence of 24 tokens.

3.4 Experimental Setup

The proposed method is compared with two baseline methods using a dataset of monophonic folk melodies in the Scottish and Irish style taken from the Session website [13]. The data comprises of monophonic melodies in ABC format[2] which are converted to MusicXML[3] using the *music21* python library [130]. For the purposes of this work, only melodies with $4/4$ time signature in which the shortest note is greater than or equal to the sixteenth note were considered, resulting in approx. 21000 melodies. The entire dataset was divided into a training set and a held-out test set using a 90%-10% split. The training set was further sub-divided into a training set and a validation set using a 70%-30% split. For training MeasureVAE, individual measures were extracted from each melody. For training the InpaintNet and the two baselines, 16 measure sequences were extracted from each melody with an overlap of 50%. All the neural network models are implemented using PyTorch.[4] Implementation details and source code are available online.[5]

[2]http://abcnotation.com, last accessed: 24th October 2020
[3]https://www.musicxml.com/, last accessed: 24th October 2020
[4]https://pytorch.org, last accessed: 24th October 2020
[5]https://github.com/ashispati/InpaintNet

3.4.1 Baseline

The performance of the proposed method is compared with the AnticipationRNN model proposed by Hadjeres et al. [116]. This model, referred to as *Base-ARNN*, uses a stack of 2 LSTM-based [131] RNN layers. Each of the 2 RNNs comprises of 2 layers with a hidden size of 256. In addition to the note-sequence tokens, this model also uses additional metadata information, i.e., tokens to indicate beat and down-beat locations as part of the user-defined constraints. For more details, the readers are directed to the AnticipationRNN paper [116].

The original model operates on tick-level sequences and inpainting locations are specified in terms of individual tick locations. Hence, the locations in the score at which inpainting is performed may or may not be contiguous. In order to make a fair comparison, a second variant of the AnticipationRNN model is considered, referred to as *Reg-ARNN*, where the stochastic training scheme from Section 3.3.2 is used instead.

3.4.2 Training Configuration

The MeasureVAE model was pre-trained using single measures following the standard VAE optimization equation with the β-weighting scheme [86, 87]. In order to prioritize high reconstruction accuracy, a low value of $\beta = 1\text{e}{-3}$ was used. Pre-training was done for 30 epochs resulting in a reconstruction accuracy of approx. 99%. While this seems to be better than results obtained by Roberts et al. [126], this could be attributed to the shorter duration of generation (single measures) and the differences in datasets and data encoding. MeasureVAE parameters were frozen after pre-training and no gradient-based updates were performed on these parameters during the InpaintNet model training.

The Adam algorithm [132] was used for model training with a learning rate of $1\text{e}{-3}$, and other optimizer hyperparameters fixed at $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1\text{e}{-8}$. To ensure consistency, all models were trained for 100 epochs (with early-stopping) with the same batch-size using a sub-sequence length of 16 measures (384 ticks). For the InpaintNet

Table 3.2: Average token-wise NLL (nats/token) on the held-out test set (lower is better). InpaintNet outperforms both baselines. The last two rows show the results for the ablation models described in Section 3.5.2.

Model Variant	**Test NLL**
Base-ARNN	0.662
Reg-ARNN	0.402
InpaintNet (Proposed)	**0.300**
PastInpaintNet	0.643
FutureInpaintNet	0.481

and Reg-ARNN models, the number of measures to be inpainted and the number of past measures were randomly selected: $n_{\mathrm{i}} \in [2, 6]$, $n_{\mathrm{p}} \in [1, 16-1-n_{\mathrm{i}}]$. The number of measures in the future context automatically becomes $n_{\mathrm{f}} = 16 - n_{\mathrm{i}} - n_{\mathrm{p}}$. This ensured that past and future contexts each contain at least 1 measure. For the baseline models, teacher-forcing [133] was used with a probability of 0.5.

3.5 Results and Discussion

In order to evaluate the effectiveness of the proposed methods, several experiments covering both objective and subjective tests were conducted. These are presented in the following sub-sections.

3.5.1 Predictions of Test Data

Two experiments were conducted to evaluate the predictive power of the models.

The first experiment considered the average token-wise negative log-likelihood (NLL) on the held-out test set. The results (see the first 3 rows of Table 3.2) indicate that the proposed model outperforms both baselines, showing an improvement of approx. 25% in the NLL over the Reg-ARNN model and approx. 55% over the Base-ARNN model.

The next experiment compared the models by varying the number of measures to be

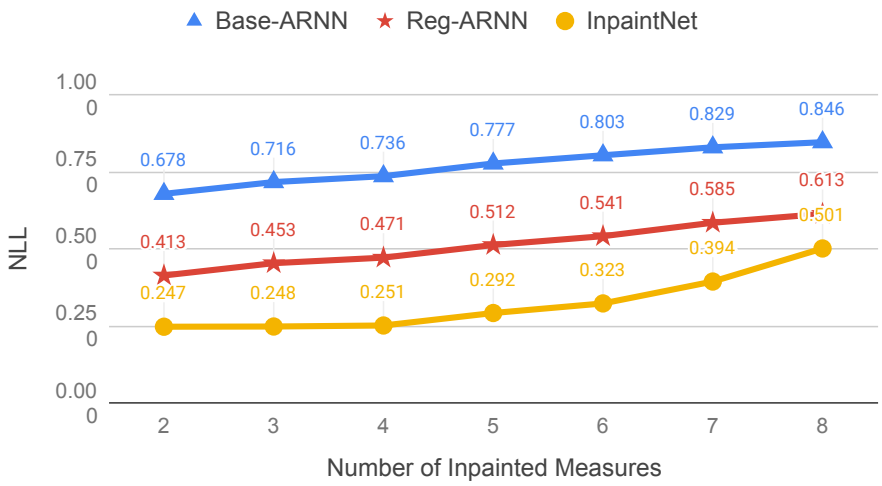


Figure 3.7: Token-wise NLL (nats/token) for different number of inpainted measures on the held-out test set (lower is better). InpaintNet outperforms both baselines. Models were trained to predict only 2 to 6 measures.

inpainted. Figure 3.7 shows the average token-wise NLL when n_{i} was increased from 2 to 8. Again, the proposed model outperforms both baselines. It should be noted that, since the sub-sequence length is constant at 16 measures, increasing n_{i} means that the available context is reduced. Thus, there is an expected drop in the performance with increasing n_{i} as the models are forced to make longer predictions with less contextual information. However, the InpaintNet model performs better even when forced to predict beyond the training limit of 6 measures.

3.5.2 Ablation Studies

In order to further ascertain the efficiency of the proposed approach, ablation studies were conducted to evaluate the benefit of adding past and future context information. Specifically, two variants of the InpaintNet model were trained which relied on only one type of contextual information. The first model, referred to as PastInpaintNet only considered the past context $\mathcal{C}_{\mathrm{p}}$ as input whereas the second model, referred to as FutureInpaintNet considered only the future context $\mathcal{C}_{\mathrm{f}}$. The last two rows of Table 3.2 summarize the performance of these ablation models. It is clear that both past and future contexts are important for the modeling

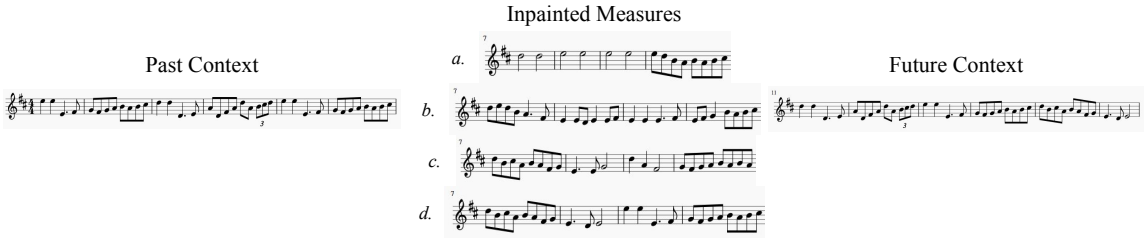


Figure 3.8: Inpaintings generated by different models for the same context. From top to bottom — *a.*: Base-ARNN, *b.*: Reg-ARNN, *c.*: InpaintNet, *d.*: Original Melody.

process.

In addition, attempts were also made to train a variant of the InpaintNet model with an untrained (randomly initialized) MeasureVAE model. However, this model failed to train properly achieving an NLL of approx. 1.33. This indicates that a latent space which is trained to learn a good representation of the data is critical for learning complex trajectories using the LatentRNN model.

3.5.3 Qualitative Analysis

Considering that the aesthetic quality of the inpaintings is of prime interest, several inpainting examples are provided online.[6] Some of those examples are used in the analysis below.

Figure 3.8 shows sample inpaintings by the models for one of the melodies in the test set. While the Base-ARNN model collapses to produce long half notes which do not effectively reflect the surrounding context, the other two models do better. Both the Reg-ARNN and InpaintNet models generate rhythmically consistent inpaintings. InpaintNet, in particular, mimics the rhythmic properties of the context better. For instance, measures 7 and 10 of the inpainted measures match the rhythm of measures 6, 14, and 15. Also, measure 8 matches measure 16. However, the use of G (subdominant scale degree in D-major) in the half-note to end measure 8 is unusual. It is also observed in other examples that the InpaintNet model occasionally produces pitches which are anomalous — either out-of-key or not fitting in the context. The Reg-ARNN model, on the other hand, tends to stay in key.

[6] https://ashispati.github.io/inpaintnet/

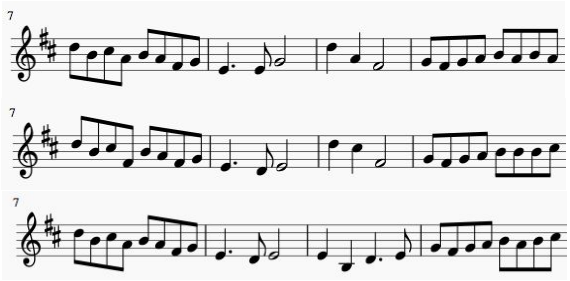


Figure 3.9: Different inpaintings generated for the same context as Figure 3.8 by the InpaintNet model.

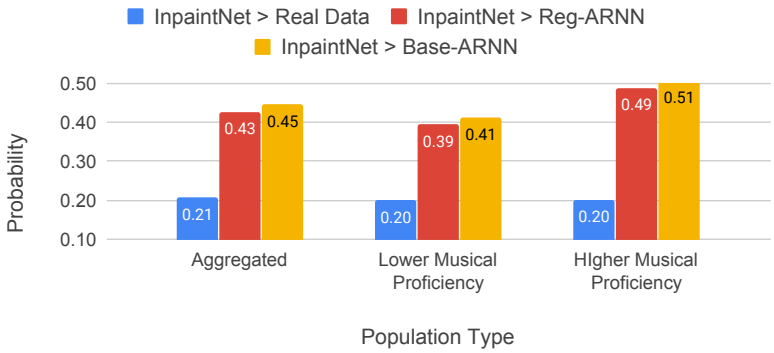


Figure 3.10: Results of the subjective listening study showing the probability that the InpaintNet model is rated higher. The analysis is based on the Bradley-Terry model [134, 135]. The proposed model loses against the real data but performs at par with the baseline models.

One advantage of working with the latent space is that the sampling operation, inherent in the VAE inference process, ensures that, given the same content, different inpainting results can be generated. Figure 3.9 shows three such generations for the context of Figure 3.8. It is interesting to note that the base rhythm is retained across all three inpaintings. This feature is particularly interesting from the perspective of compositional tools, as this model can be used to quickly provide users with multiple ideas for the same context which will allow greater interactivity.

3.5.4 Subjective Listening Test

To evaluate the perceived quality of the inpainted measures, a listening test was conducted to compare the proposed model against the two baselines. A set of 30 melodies from the

held-out test set were randomly selected and their first 16 measures were extracted. The models were then used to inpaint 4 measures (measure number 7 to 10) in these melodic excerpts. Participants were presented with pairs of melodic excerpts and asked to select the one in which they thought the inpainted measures fit better within the surrounding context. In some of the pairs, one melodic excerpt was the real data (without any inpainting). Each participant was presented with 10 such pairs. A total of 72 individuals participated in the study (720 comparisons). The location of the inpainted measures was kept consistent across all examples so as to prevent confusion among participants and allow them to focus better on the inpainted measures.

The Bradley-Terry model [134, 135] for paired comparisons was used to get an estimate of how the proposed model performs against the baselines and the real data (see Figure 3.10). While the proposed model expectedly has a very low probability of winning against the real data (wins approx. 1 out of 5 times), it performs only at par with the baseline models (with probability approx. 0.5). Significance tests using the Wilcoxon signed-rank test were further conducted which validated that differences between the proposed model and the baselines were not statistically significant (p-value > 0.01). This was unexpected since the proposed model showed significant improvement over the baselines in the NLL metric. Further dividing the study population into two groups differing in musical proficiency (based on the Ollen index [136]) showed that, comparatively, the group with greater musical proficiency favored the generations from the InpaintNet model more than the group with less musical proficiency.

Additional analysis revealed that cases where the InpaintNet model performed the worst (maximum losses against the baselines), had anomalies in the predicted pitch similar to those discussed in Section 3.5.3. Specifically, they either had a single out-of-key note (e.g., F note in G-Major scale) or used a pitch or interval not used in the provided contexts. It is conjectured that these anomalous pitch predictions lead to poor perceptual ratings in spite of the model performing better in terms of modeling rhythmic features. This imbalance of

pitch versus rhythm predictions on perceptual ratings is something which can be analyzed further in future studies.

3.6 Conclusion

This chapter investigated the problem of musical score inpainting and proposed a novel approach to generate multiple measures of music to connect two musical excerpts by using a conditional RNN which learns to traverse the latent space of a VAE. It also improved upon the data encoding scheme and introduced a stochastic training process which facilitates model training and improves generalization. The experimental results show that learning methods can be used to leverage the high-level musical information (such as rhythm, musical structure, and repetition) encoded in the latent spaces to accomplish challenging music generation tasks. Comparisons with the baseline architectures, which directly operate on token level sequences, also show that operating on latent vector sequences is more advantageous. Recent work on learning more meaningful representations for polyphonic music also seems to align with this finding [137].

The proposed architecture provides users with several interesting options for interaction. Based on their aesthetic sensibilities and compositional goals, users can choose from the multiple inpainting suggestions provided by the model for the same context. The stochastic training scheme ensures that the model can perform inpaintings at arbitrary locations. This provides the users the option to fix certain measures that they like and ask the model to suggest inpaintings for the other measures. Users can also tweak parts of the past and future contexts and then use the model to fill in the gap. These features allow the model to be used in the back-end of interactive music creation interfaces [72, 138].

The idea of learning to traverse latent spaces could be useful for other music generation tasks also. For instance, the architecture of the LatentRNN model can be changed to add contextual information from other voices/instruments to perform multi-instrument music generation. Another promising avenue for future work is substituting RNNs with attention-

based models [139] which have had success in sequential music generation tasks [16]. This, in turn, has the potential to be used to generate music with long-term structure.

CHAPTER 4
LATENT SPACE REGULARIZATION

The previous chapter presented a learning method to leverage the encoded information in VAE latent spaces to perform interactive music generation tasks. However, one of the key limitations of the vanilla-VAE framework is that the encoded attributes in the latent space cannot be explicitly controlled and the learnt attributes are often not interpretable by humans. In order to circumvent this limitation, there has been substantial research on modifying the VAE training procedure to learn representations which are able to disentangle different data attributes using either unsupervised [86, 92, 91, 114] or supervised [100, 67, 66, 101, 83] methods. The limitations associated with both categories of methods have been discussed in Section 2.2.

This chapter presents the Attribute-Regularized VAE (AR-VAE) framework which uses a supervised training method to create structured latent spaces where specific attributes are forced to be encoded along specific dimensions of the latent space. In order to achieve this, a novel regularization loss is formulated which forces each specific attribute of interest to have a monotonic relationship with the latent code of the dimension along which the attribute is encoded (hereafter referred to as the *regularized dimension*). Fig. 4.1 demonstrates this overall idea. The proposed method can be used to learn disentangled latent representations which could be used for manipulating different data attributes. The superior performance of AR-VAE compared to the baseline models is demonstrated using several quantitative and qualitative experiments.

The rest of this chapter[1] is organized as follows: firstly, the AR-VAE method is described in Section 4.1. Next, the details of the experimental set-up are discussed in Section 4.2. The results of the different quantitative and qualitative experiments conducted are presented in

[1]Parts of this chapter have been published in [71] and [68].

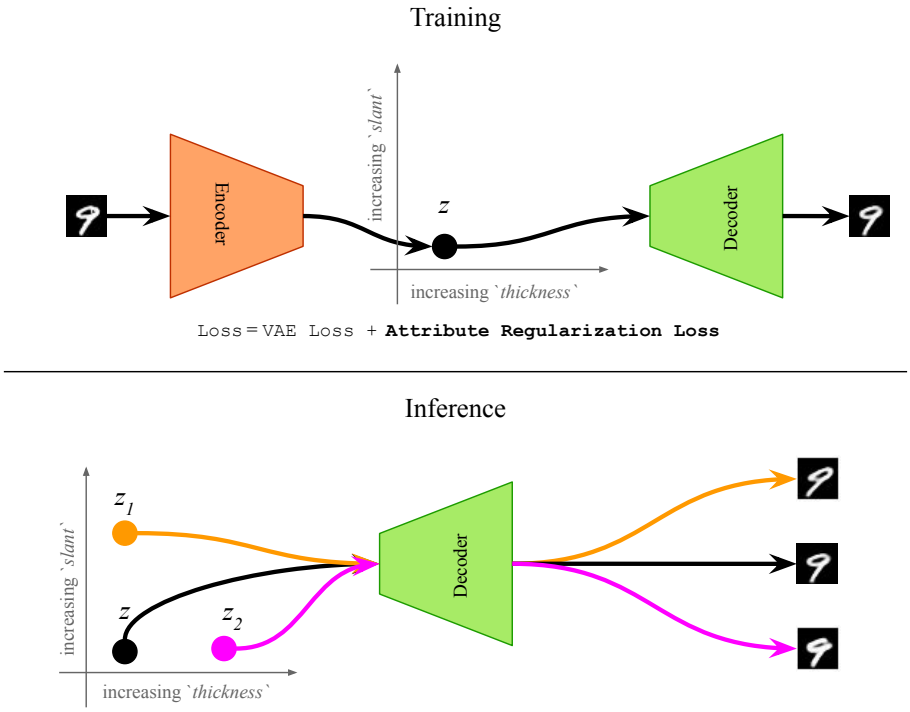


Figure 4.1: Motivation for the AR-VAE model which uses a novel attribute regularization loss (see Section 4.1.1) during the training step to force the latent space to encode specific attributes along specific dimensions of the latent space of a VAE. During inference, individual data attributes can be manipulated by simply traversing along these regularized dimensions.

Section 4.3. Finally, Section 4.4 concludes the chapter.

4.1 Method

The goal of the proposed method is to train a structured latent space in which individual attributes are encoded along specific dimensions of the latent space. Such a latent structure enables controllable generation by selectively modifying the latent code of the regularized dimension. For instance, if the attribute represents 'thickness' of a digit, and the regularized dimension corresponds to the first dimension of the latent space, then sampling latent vectors with increasing values of the first dimension should result in digits with increasing thickness as illustrated in Figure 4.1.

A brief background on VAEs was already provided in Section 2.1.1. The subsequent

parts of this section describe the regularization loss formulation and the learning algorithm for AR-VAE.

4.1.1 Attribute Regularization Loss

Consider a $\mathbb{D}$-dimensional latent space where latent vectors are represented as $\mathbf{z} : \left\{z^k\right\}, k \in [0, \mathbb{D})$. Formally, the objective is to encode an attribute a along a dimension r of the latent space such that, traversal along r should result in an increase in the attribute value a of the generated data. Note that the attribute values must be continuous (or at least ordered). Mathematically, if $a(\mathbf{x}_i) > a(\mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two data-points generated using latent vectors $\mathbf{z}_i$ and $\mathbf{z}_j$, then $z_i^r > z_j^r$ should hold for any arbitrary i and j.

This is accomplished by adding an attribute-specific regularization loss to the VAE training objective. To compute this loss, a mini-batch containing m training examples is used, and a three-step process is followed:

(a) An attribute distance matrix $D_a \in \mathbb{R}^{m \times m}$ is computed for all examples in the training mini-batch:

$$D_a(i, j) = a(\mathbf{x}_i) - a(\mathbf{x}_j) \tag{4.1}$$

where $i, j \in [0, m)$.

(b) Next, a similar distance matrix $D_r \in \mathbb{R}^{m \times m}$ is computed for the regularized dimension r of the latent vectors:

$$D_r(i, j) = z_i^r - z_j^r. \tag{4.2}$$

(c) The regularization loss is finally formulated as:

$$L_{r,a} = \mathrm{MAE}(\tanh(\delta D_r) - \mathrm{sgn}(D_a)), \tag{4.3}$$

where $\mathrm{MAE}(\cdot)$ is the mean absolute error, $\tanh(\cdot)$ is the hyperbolic tangent function, $\mathrm{sgn}(\cdot)$ is the sign function, and δ is a tunable hyperparameter which decides the

degree to which the encoded points spread in the latent space.

The $\mathrm{sgn}(\cdot)$ function is used with the attribute distance matrix D_a since we are only interested in whether the attribute value of a data-point is higher or lower than the others in the mini-batch and do not care about the magnitude of the differences. The $\tanh(\cdot)$ function is used for the regularized dimension distance matrix D_r since it has the same range as $\mathrm{sgn}(D_a)$, i.e, $[-1, 1]$, and we want to minimize the mean absolute error between $\tanh(\delta D_r)$ and $\mathrm{sgn}(D_a)$. In addition, $\tanh(\cdot)$ is a differentiable function which ensures that the loss is differentiable with respect to the latent vectors (and consequently the encoder parameters). Thus, gradient descent can be used for optimization. Other functions having such properties could also be potentially used to replace $\tanh(\cdot)$.

Overall, this formulation forces the latent code of the regularized dimension to have a monotonic relationship with the attribute values. Note that, unlike the GLSR-VAE formulation [67] which requires that the attributes should be computed using differentiable functions, this formulation is agnostic to the way in which the attributes are computed/obtained.

4.1.2 Learning Algorithm

If the attribute set $\mathcal{A} : \{a_l\}, l \in [0, \mathbb{L})$, contains $\mathbb{L}$ attributes, ($\mathbb{L} \leq \mathbb{D}$), then the overall loss function for AR-VAE is formulated as:

$$L_{\text{AR-VAE}} = L_{\text{recons}} + \beta L_{\text{KLD}} + \gamma \sum_{l=0}^{\mathbb{L}-1} L_{r_l,a_l} \tag{4.4}$$

where L_{r_l,a_l} is the regularization loss for the attribute a_l with r_l as the index of the regularized dimension, and γ is a tunable hyperparameter which is referred to as the regularization strength. For brevity, symbols θ and ϕ corresponding to the network parameters are omitted in the above equation. The overall learning algorithm for AR-VAE is shown in Algorithm 1.

Algorithm 1: Learning algorithm for AR-VAE

Input: observations and attribute labels $(\mathbf{x}_i, \{a_l\}_i)_{i=1}^N$ (N is the number of examples in the dataset, $l \in [0, \mathbb{L})$ where $\mathbb{L}$ is the number of attributes), batch-size m, indices of the latent dimensions to be regularized $\{r_l\}_{l=0}^{\mathbb{L}-1}$, initialized VAE encoder and decoder parameters ϕ, θ, neural network optimizer g

repeat

  Randomly sample a batch of m data-points $(\mathbf{x}_i, \{a_l\}_i)$

  Compute $L_{\text{recons}}(\theta, \phi)$ using Equation 2.3

  Compute $L_{\text{KLD}}(\theta, \phi)$ using Equation 2.4

  for $l \in [0, \mathbb{L})$ **do**

    Compute L_{r_l, a_l} using Equation 4.3

  end

  Compute $L_{\text{AR-VAE}}(\theta, \phi)$ using Equation 4.4

  Update VAE encoder and decoder parameters: $\theta, \phi \leftarrow g(L_{\text{AR-VAE}}(\theta, \phi))$

until *convergence of objective*

4.2 Experimental Setup

This section presents details regarding the experimental set-up used to evaluate the AR-VAE framework. A description of the different datasets and attributes used, the baselines considered, the model architectures, and other implementation details are presented.

4.2.1 Datasets and Attributes

The performance of the AR-VAE framework is evaluated using two music-based datasets described below. In order to also investigate if the AR-VAE framework is general enough to be applied to other domains, a couple of common image-based datasets are chosen.

Image Datasets and Attributes: The first dataset from the image domain is the dSprites dataset which contains approximately 0.7 million two-dimensional shapes having 5 simple factors of variation: *shape*, *scale*, *orientation*, *x-position*, and *y-position* [64]. This is a standard dataset for evaluating disentanglement methods. The second dataset is the Morpho-MNIST dataset which contains 70000 handwritten MNIST digits along with complex morphological attributes for each digit obtained using computational methods [140]. The

attributes are *area*, *length*, *thickness*, *slant*, *width*, and *height*.

Music Datasets and Attributes: For evaluating the performance of AR-VAE on music data, two datasets consisting of single measures (bars) of monophonic melodies are used. The first dataset consists of measures extracted from the soprano parts of the J.S. Bach Chorales dataset [130] (≈ 350 chorales). The second dataset consists of measures extracted from approximately 20000 folk melodies in the Scottish and Irish style [141]. Since pitch and rhythm are the two primary features of a monophonic melody, the following four attributes are considered for both datasets:

(a) *note density*, the count of the number of notes in each measure,

(b) *pitch range*, the difference of lowest pitch value (in MIDI) in the measure from the highest pitch value,

(c) *rhythmic complexity*, based on Toussaint's metrical complexity measure [142], and

(d) *contour*, the degree to which the melody moves up or down measured by summing up the difference in pitch values of all the notes in the measure.

These chosen attributes cover important aspects of short monophonic melodic sequences. The first two attributes have been used in previous studies on controllable melody generation [67, 66]. The metrical complexity measure used to compute the third attribute has been shown to have some correlation with human perception of rhythmic complexity [142]. Finally, the *contour* attribute represents the overall direction of the melodic movement which also has some perceptual relevance. All the attributes are computed using the steps described in Appendix A.1.

4.2.2 Baselines

Most of the unsupervised methods for disentanglement learning have been shown to perform at par with one another [89]. Since AR-VAE improves upon the β-VAE model [86] (compare

Equation 2.5 to Equation 4.4), the latter is chosen as a baseline for comparison. It also doubly serves as an ablation case. In addition, the β-VAE model is also shown to perform at par with some supervised models [86] and hence, can be considered as a suitable baseline.

Other supervised methods such as GLSR-VAE [67] and Fader networks [100] were also considered as potential baselines. However, the former requires differentiable computation of attributes (which cannot be applied to the datasets and attributes considered in this paper), and the latter is designed for binary/categorical attributes. Attempts were made to adapt the Fader network design to work with continuous attributes. However, it did not lead to competitive results. A recent study by Locatello et al. proposed a different regularization formulation [102] for supervised disentanglement with limited labels. This regularization loss normalizes the attributes within $[0, 1]$ and uses a binary cross-entropy loss to match attribute values to the regularized dimension. The resulting model, referred to here as S2-VAE, is considered for comparison.

4.2.3 Implementation Details

Different model architectures are chosen for different data domains. Convolutional architectures are used for the VAEs trained on images whereas recurrent architectures are used with the music-based datasets. To ensure consistency, all models for a particular dataset are trained for the same number of epochs using the same optimizer and learning rate. For the supervised methods (AR-VAE and S2-VAE), the models are trained to regularize all the attributes for any given dataset. The model architectures and other details are provided in Appendix A.2.[2]

For training the models, each dataset is divided into a train-validation-test split using an 80%-15%-5% ratio. The train and validation sets are used to train the models and tune the hyperparameters. The held-out test sets are used for evaluation. For each dataset, all models are trained for 10 different random initializations. Based on initial experiments, for the

[2] Also provided online at: https://github.com/ashispati/ar-vae

β-VAE models, β is chosen as 4.0 and $1\mathrm{e}{-3}$ for the image and music datasets, respectively (the low value of β was necessary to train the music VAE models while maintaining a high reconstruction accuracy [70, 11]). For AR-VAE, the models for the image datasets are trained with $\gamma = 10.0$ and $\delta = 1.0$, whereas those for the music datasets are trained with $\gamma = 1.0$ and $\delta = 10.0$. For S2-VAE, the same value of γ is used as for the corresponding AR-VAE model.

4.3 Results and Discussion

Several quantitative and qualitative experiments are conducted to comprehensively evaluate the performance of the AR-VAE framework:

(a) First, the disentanglement performance of AR-VAE is evaluated across all the datasets by using different objective metrics.

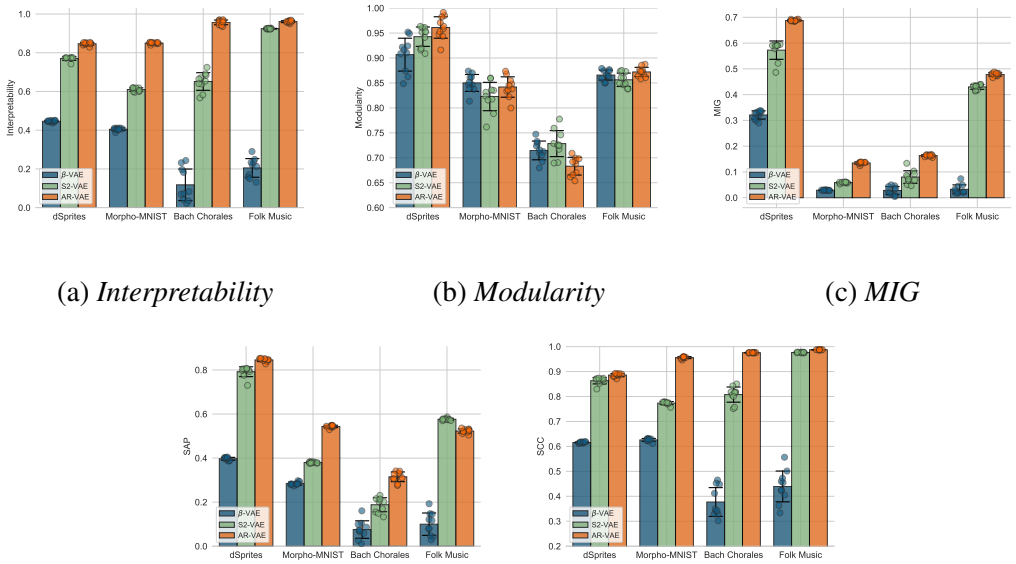
(b) Second, the reconstruction fidelity of AR-VAE is compared against other models by using both reconstruction accuracy as well as inspecting the quality of the reconstructions.

(c) Third, an experiment using the Folk Music dataset objectively measures how well AR-VAE is able to disentangle individual attributes during data generation.

(d) Fourth, qualitative inspections of generated data is used to gain additional insights into the degree of controllability provided by AR-VAE.

(e) Fifth, the overall structure of the latent space with respect to the different attributes is investigated by using attribute-specific visualizations plots. This is done both from the perspective of the AR-VAE encoder and decoder.

(f) Sixth, an experiment using the Morpho-MNIST dataset is conducted which objectively evaluates the degree to which the overall content of the generated data is preserved while trying to manipulate different attributes.



(a) *Interpretability* (b) *Modularity* (c) *MIG*

(d) *SAP* (e) *SCC*

Figure 4.2: Bar plots for disentanglement performance for different methods across different datasets (higher is better). The circular dots denote results for each random seed. Error bars correspond to one standard deviation.

(g) Finally, the sensitivity of the AR-VAE framework to the two hyperparameters is evaluated by using an objective study on the Morpho-MNIST dataset.

For all experiments involving data generation, simple traversals along different regularized dimensions are used. Specific details are provided in the individual experiments described below.

4.3.1 Disentanglement

This experiment deals with evaluating the degree of disentanglement of the latent space with respect to the different data attributes. The different metrics introduced in Section 2.4 are used for evaluation. Figure 4.2 shows the performance of the models across all datasets and metrics. The following observations can be made:

(a) AR-VAE clearly outperforms both baselines across all metrics (except *Modularity*). Moreover, this superior performance extends across all datasets in spite of the different

domains and the varying degree of complexity in the attributes.

(b) Compared to β-VAE, AR-VAE shows considerable improvement for the *Interpretability* metric and the *SCC* (Spearman Correlation Coefficient). This is expected since AR-VAE forces a monotonic relationship between a given attribute and the latent code of the regularized dimension. AR-VAE also outperforms S2-VAE across these two metrics.

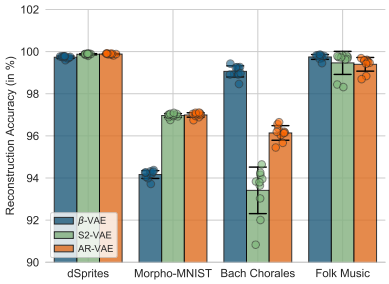
(c) There is a lower improvement over the baselines for *MIG* (Mutual Information Gap) which would suggest that there are other dimensions (apart from the regularized dimension) which share high mutual information with different attributes. This would also explain why the *SAP* (Separated Attribute Predictability) score does not improve as much as the *Interpretability* metric.

(d) For the image-based datasets, the metrics for dSprites are generally higher than the morpho-MNIST dataset. This could be due to the artificial nature of the former and its simpler attributes. For the music-based datasets, Folk music has better performance than Bach chorales. This might be due to the significantly larger size of the Folk dataset.

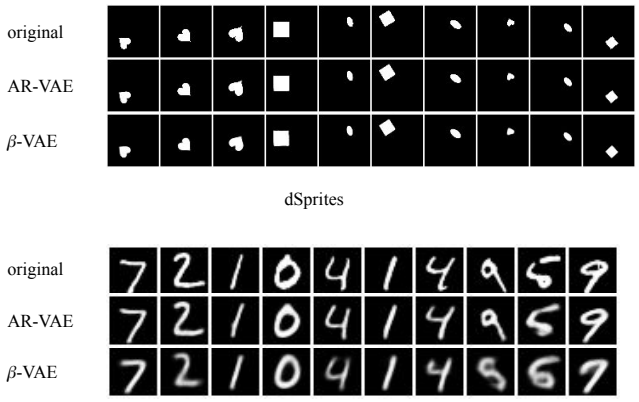
4.3.2 Reconstruction Fidelity

The reconstruction quality is another important criterion. While a high degree of disentanglement is desirable, it should not be at the cost of a drastic decrease in the quality of the generation.

Figure 4.3(a) shows the reconstruction accuracies. While the performance of the three models is in the same range for the dSprites dataset, the supervised methods (AR-VAE and S2-VAE) perform better for the Morpho-MNIST dataset. A few example reconstructions for the dSprites and Morpho-MNIST datasets are shown in Figure 4.3(b). The better performance of AR-VAE can be gauged from the sharper reconstructions of MNIST digits.



(a) *Reconstruction Accuracy*



(b) *Reconstruction Examples*

Figure 4.3: Reconstruction results for AR-VAE compared to other models. (a) Bar plots of reconstruction accuracies for different methods across different datasets (higher is better). (b) Example reconstructions for the image-based datasets. The AR-VAE model has sharper reconstructions compared to the β-VAE model for the morpho-MNIST dataset.

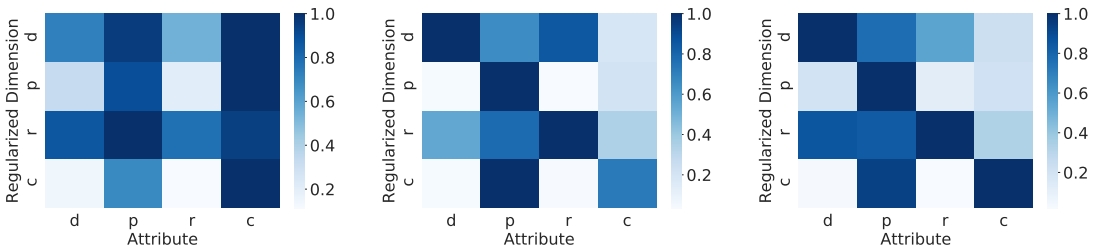
S2-VAE has a similar reconstruction performance as AR-VAE.

For the music datasets, there is a drop in reconstruction accuracy for the supervised methods. A slight drop in the case of the Folk dataset is expected since the same value of β is used for all three models. Thus, supervised models have more constraints than β-VAE during training. The larger drop in performance for the Bach Chorales might be due to the smaller size of the dataset. Note that AR-VAE has a slightly better reconstruction performance than S2-VAE on the smaller Bach Chorales dataset.

4.3.3 Attribute Disentanglement during Generation

While the disentanglement results in Section 4.3.1 are promising, they are computed using data from the held-out test set only. It is even more important to look at how effective the disentanglement is when new data is being generated. Specifically, as new data is generated by traversing along a specific regularized dimension, only the corresponding attribute should change.

To measure this quantitatively, the following procedure is used. Given a data-point



(a) β-VAE (b) S2-VAE (c) AR-VAE

Figure 4.4: Net change in attribute values during traversal along the regularized dimensions for the Folk Music dataset. The columns show the normalized net change in a particular attribute value as one traverses the regularized dimension for the attribute corresponding to the rows. d: *note density*, p: *pitch range*, r: *rhythmic complexity*, c: *contour*.

with latent vector $\mathbf{z}$, 5 different variations are generated by uniformly interpolating along the dimension r_l from -4 to 4, where r_l is the regularized dimension for attribute a_l. For the β-VAE model, the dimension with the highest mutual information with the attribute is considered as the regularized dimension. An attribute change matrix $A \in \mathbb{R}^{L \times L}$, where L is the number of attributes, was computed using the following formulation:

$$A(m, n) = \sum_{i=1}^{4} |a_n(\mathbf{z}_i^m) - a_n(\mathbf{z})| \tag{4.5}$$

where, $A(m, n)$ computes the net change in the n^{th} attribute as one traverses the dimension r_m (which is the regularized dimension for the m^{th} attribute), $a_n(\cdot)$ is the value of the n^{th} attribute, and $\mathbf{z}_i^m$ is the i^{th} interpolation of $\mathbf{z}$ obtained by traversing along the r_m dimension. This attribute change matrix is computed for each model type by averaging over a total of 1024 data-points in the test-set and across all 10 random seeds. The matrix is also normalized so that the maximum value across each row corresponds to one.

The results obtained for the Folk Music dataset (results on the Bach Chorales were similar) are shown in Figure 4.4. Ideal disentanglement should result in a plot with high values (dark blue color) along the diagonal (as there should be a change in the regularized attribute only) and lighter colors on the off-diagonal entries. It is clear that among the three

different models, AR-VAE is the closest to this ideal behavior. β-VAE, on the other hand, performs the worst. The following additional observations can be made:

(a) For the β-VAE model, the latent space seems to be highly entangled. Traversing along any regularized dimension leads to a large change in the *contour*. Also, traversing along the dimension regularizing *rhythmic complexity* leads to greater change in *pitch range*.

(b) S2-VAE shows pretty good performance overall. However, there is a greater change in *pitch range* when traversing along the dimension regularized for *contour* which is undesirable.

(c) AR-VAE also suffers from the same confusions (e.g, 'd' and 'p' changes while traversing along 'r'). However, the degree of change across a row is always highest for the regularized attribute which is desirable.

(d) All the models struggle to disentangle *note density* and *rhythmic complexity*. This might be due to a high (Spearman's) correlation between the two attributes (≈ 0.89). This is also reflected in the relatively poor *Interpretability* score (in the experiment conducted in Section 4.3.1) for *Rhythmic Complexity* (0.83) compared to the other attributes (≈ 0.99). This is discussed in greater detail in the next chapter (see Section 5.3.3).

4.3.4 Inspecting Latent Interpolations

This experiment presents a qualitative evaluation of the degree to which AR-VAE provides control over individual data attributes during the generation process. The same process as the previous Section 4.3.3 is followed except that 10 different interpolations are generated for the image-based datasets. The results are compared to β-VAE.

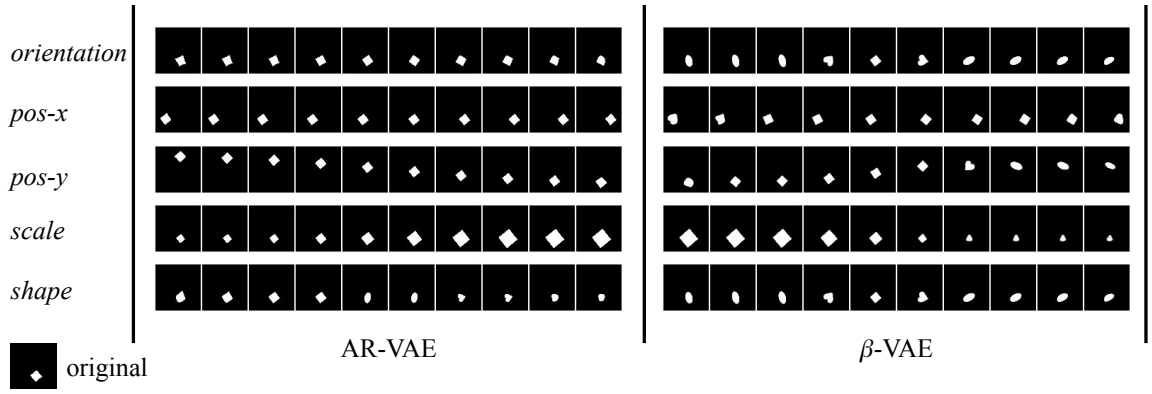


Figure 4.5: Controlling different attributes of a dSprites image (shown in the bottom left). AR-VAE model is able to manipulate each attribute independently. However, while the β-VAE model is also able to separately control *pos-x*, *pos-y*, and *scale*, it changes the *shape* of the sprite while manipulating *orientation* and vice versa.

Images: Figure 4.5 and Figure 4.6 show the results of controlling attributes for the dSprites and the Morpho-MNIST datasets, respectively. While both models are able to control the individual attributes to a similar degree for the dSprites dataset (Figure 4.5), the AR-VAE model performs better for the Morpho-MNIST dataset (Figure 4.6). Not only are the interpolations meaningful with respect to the regularized attribute, but AR-VAE is able to retain the identity of the original digit in most cases while β-VAE fails to do so (Sect. 4.3.6 further supports this observation). Additional results are provided in Appendix A.3.1.

Music: Figure 4.7 shows the results of manipulating different musical attributes in the Bach Chorales and Folk Music datasets. For AR-VAE, in most cases, traversal along the regularized dimensions lead to measures with increasing values of the attributes. These can be seen more clearly in the attribute value plots to the right of the piano-rolls.

On the contrary, for β-VAE, the progression of the attributes is not as uniform. In fact, in the Bach Chorales example, there is no change in the values for two out of the four attributes. This is because in the case of β-VAE, on many occasions, there is a single dimension of the latent space which has maximum mutual information with two or more attributes. This could be either due to a high degree of correlation between attributes (e.g., *Rhythmic*

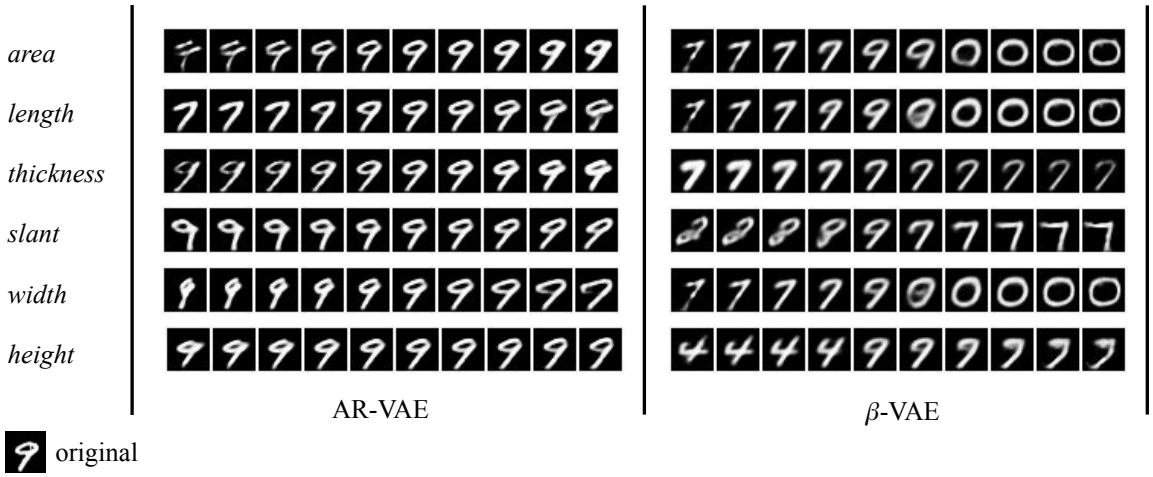
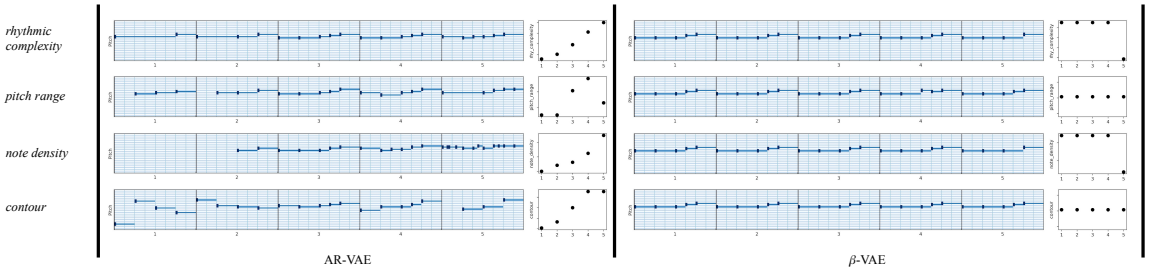


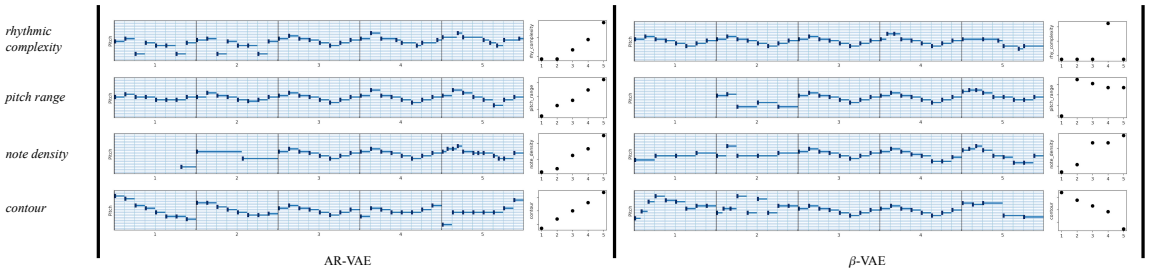
Figure 4.6: Controlling different attributes of a Morpho-MNIST digit (shown in the bottom left). The AR-VAE model is able to separately control the individual attributes of the original image and retains the identity of the digit in most cases. The β-VAE model, on the other hand, fails to retain the identity of the original digit and the interpolations are not meaningful.

Complexity and *Note Density*) or due to very poor disentanglement where all attributes are poorly encoded. Even in cases where the attributes have a monotonic progression (e.g., see the bottom right row for the *Contour* attribute in the Folk Music examples in Figure 4.7), the order of the encoding is reversed (the attribute decreases with increasing value of the latent code). This warrants a post-hoc analysis of the latent space to understand the relationship between the attributes and the latent dimensions.

It is important to point out that while AR-VAE allows better control over the different attributes for both datasets, it sometimes struggles to maintain the musical coherency of the generated measures. For instance, the generated measures shown in Figure 4.7 are not always in the same key. This is clearer in Figure 4.8 which shows the corresponding musical scores. Although this seems to be the case for measures generated with the β-VAE model as well, the problem seems to be more pronounced for AR-VAE. It was observed that training separate models for the different attributes, where each model regularizes a single attribute, tends to alleviate this limitation. Traversing along the regularized dimensions in these models typically results in better musical coherence.



(a) Bach Chorales



(b) Folk Music

Figure 4.7: Controlling different attributes of musical measures. The piano-rolls show measures generated by increasing the latent code of the regularized dimension (or in the case of β-VAE, the dimension with the highest mutual information) for the respective attribute. The light vertical lines within each measure denote the location of the eighth-notes. The y-axis of the piano rolls shows pitch in semi-tones. The plots on the right show how the attribute values change with the increase in the latent code. For AR-VAE, in most cases, traversing along the regularized dimension leads to an increase in attribute values.

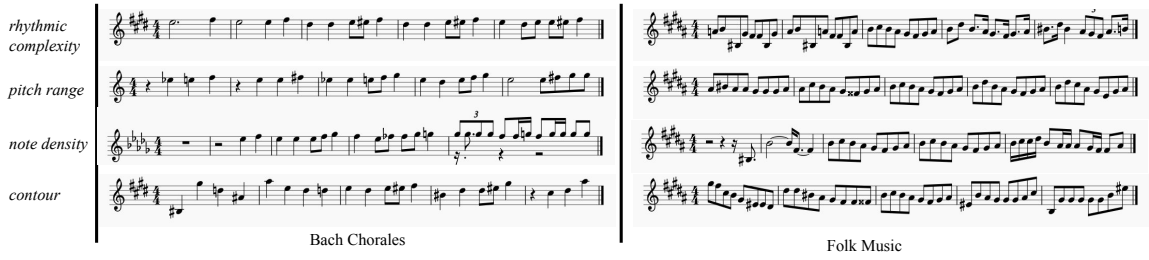
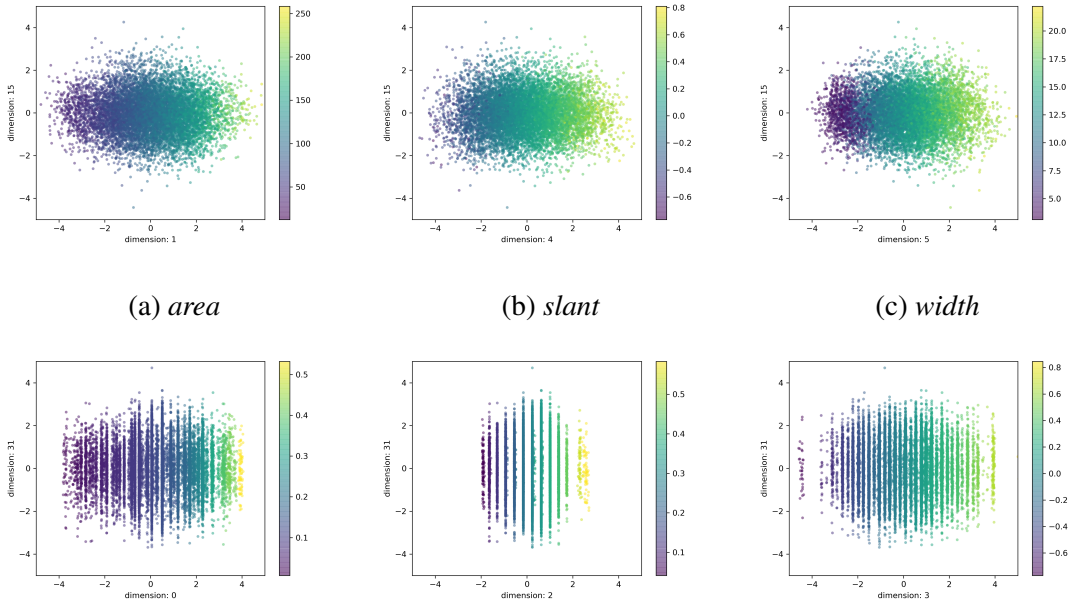


Figure 4.8: Musical score corresponding to the AR-VAE generated interpolations from Fig. 4.7. While the attribute values of the generated measures are controlled effectively, the musical coherence is often lost (particularly in the case of Bach Chorales).



(a) *area* (b) *slant* (c) *width*

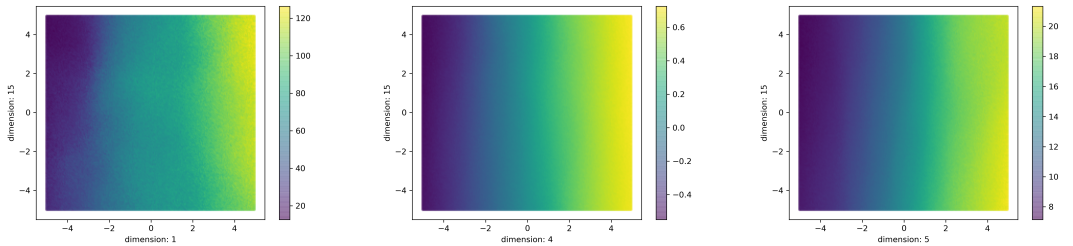
(d) *rhythmic complexity* (e) *note density* (f) *contour*

Figure 4.9: Encoder data distribution plots for selected attributes. The top row shows three attributes from the Morpho-MNIST dataset, the bottom row shows three attributes from the Folk Music dataset. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Lighter colors indicate higher attribute values.

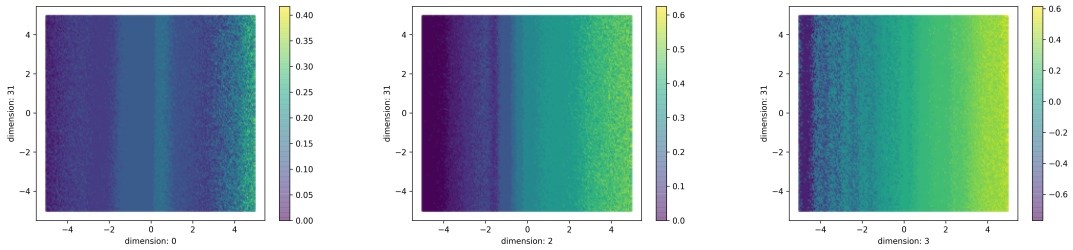
4.3.5 Latent Space Visualization

In this experiment, the structure of the latent space with respect to different attributes is investigated by using 2-d visualizations. First, the performance of the AR-VAE encoder is considered by plotting how the encoded data is distributed in the latent space. This gives insights into how well the regularization process works. Next, the performance of the AR-VAE decoder is evaluated by considering latent surface plots. These show how well the decoder is able to control the attributes of the generated data. The details regarding the generation of these plots are described below.

For the data distribution plots, first, latent representations are obtained for the data from the held-out test sets by using the AR-VAE encoder. Then, for each attribute, these representations are projected onto a 2-dimensional plane where the x-axis corresponds to the



(a) *area* (b) *slant* (c) *width*



(d) *rhythmic complexity* (e) *note density* (f) *contour*

Figure 4.10: Decoder latent surface plots for selected attributes. The attributes shown are the same as in Figure 4.9. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Lighter colors indicate higher attribute values.

regularized dimension and the y-axis corresponds to a non-regularized dimension. Figure 4.9 shows the results for a few selected attributes each from the Morpho-MNIST and Folk Music datasets. It is clear that the AR-VAE encoder is able to effectively encode data-points so as to maintain a monotonic relationship between the attribute values and the corresponding regularized dimension. This can be seen from the gradual transition from purple to yellow colors along the regularized dimension for the encoded data-points. No such change in color is seen along the non-regularized dimension.

To show how effectively the AR-VAE decoder is able to control different attributes during the generation process, latent surface plots are used. For a given attribute, a 2-dimensional plane on the latent space is considered which comprises of the regularized dimension (x-axis) for the attribute and a non-regularized dimension (y-axis). The latent code for the other dimensions is drawn from a normal distribution and kept fixed. The latent

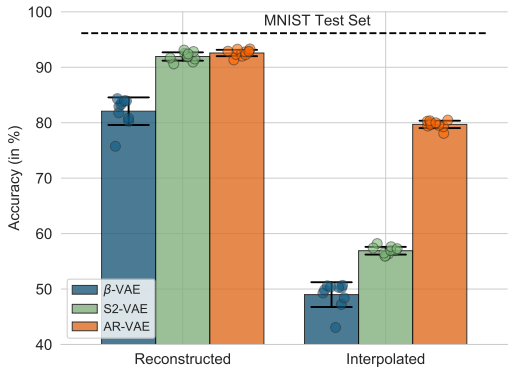


Figure 4.11: Distribution of accuracy (higher is better) in predicting the MNIST test set digits. Higher values indicate the greater ability of the model in preserving the image content. 'Reconstructed': prediction accuracy on reconstructed images from the MNIST test set, 'Interpolated': prediction accuracy on interpolations generated by traversing along the regularized dimensions for different attributes. The prediction accuracy on the original MNIST test set is shown with a dashed line.

vectors thus obtained are passed through the AR-VAE decoder and the attributes of the generated data are computed. Figure 4.10 shows the result of this visualization for a few selected attributes (same as in Figure 4.9). In most cases, the attributes show an increasing trend as the latent code of the regularized dimension is increased. This is seen by the gradual transition from purple to yellow color along the x-axis. The change in color is minimal for the y-axis which indicates independence of the attribute from the non-regularized dimension. The only attribute for which this does not hold is *rhythmic complexity*. Once again, this could be on account of its high correlation with *note density*. The choice of the attributes and their relationship to each other, thus, becomes an important consideration to create meaningful latent spaces.

Overall, the AR-VAE latent space is interpretable with respect to the attribute values and traversing along the regularized dimension leads to an increase in the corresponding attribute. Additional results for all the attributes are shown in Appendix A.3.2.

4.3.6 Content Preservation

In order to ascertain AR-VAE's ability to retain the content during interpolation, an additional experiment is conducted using the morpho-MNIST dataset. The identity of the digit is used as a proxy to the image content. Different variations of an input digit are generated by changing the attributes (via appropriate manipulation of the latent code) and then a pre-trained model is used to predict the digit class. The higher the classification accuracy, the better the model is at preserving the identity of the input. The experiment is run using the 10000 digits in the MNIST test set. For each digit, and for each model, 60 variations are generated by interpolating along the regularized dimensions (10 interpolations each for the six attributes). For β-VAE, the dimension with the highest mutual information with an attribute is considered as the regularized dimension. The pre-trained model has a ResNet-based architecture with an accuracy of 96.15% on the unmodified MNIST test set. The results of this experiment are shown in Figure 4.11. While there is only a small difference in the performance of the three models in retaining the identity of the reconstructed digits, AR-VAE significantly outperforms both baselines for the interpolations. This indicates that AR-VAE is better for manipulating attributes while retaining the underlying content.

4.3.7 Hyperparameter Sensitivity

The next experiment assesses the sensitivity of the AR-VAE regularization to the two hyperparameters (γ and δ). Figure 4.12 shows the trade-off between reconstruction accuracy and the *Interpretability* metric as the hyperparameters are varied for the Morpho-MNIST dataset while keeping β fixed at 1.0.

For lower values of γ, an increase in δ results in only marginal improvements in the *Interpretability* metric without any loss in reconstruction accuracy. However, after γ crosses 1.0, increasing δ leads to substantial improvement in the *Interpretability* metric accompanied by a slight drop in the reconstruction accuracy. Note that the β-VAE model (shown with a $\times$) performs considerably worse in comparison. Choosing γ in the $[5.0, 10.0]$ range and δ in

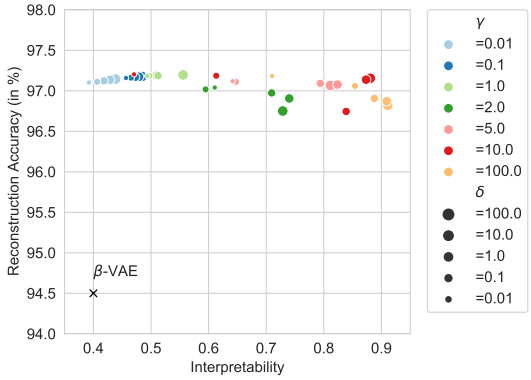


Figure 4.12: Effect of the hyperparameters γ and δ on the performance of the AR-VAE model (β is fixed at 1.0) on the Morpho-MNIST dataset. Each dot corresponds to a unique combination of γ and δ which are indicated by the color and the size of the dots respectively. The β-VAE model ($\beta = 4$) is also shown (with a $\times$) for reference. The ideal model should lie on the top right corner of the plot (with high values of both reconstruction accuracy and *Interpretability* metric).

the $[1.0, 10.0]$ range seems to give the best results.

4.4 Conclusion

This chapter investigated the problem of selective manipulation of data attributes in deep generative models by structuring the latent space of a VAE to encode specific attributes along specific dimensions of the latent space. The proposed AR-VAE model uses a novel regularization loss to enforce a monotonic relationship between the attributes and the latent code of the respective regularized dimension. The resulting latent spaces are easily interpretable and allow manipulation of individual attributes by simple traversals along the regularized dimensions. The regularization loss works for continuous data attributes and has a simple formulation with only two hyperparameters. In addition, contrary to previous supervised methods [67, 83], the loss formulation is agnostic to how the attributes are computed or obtained. Using both image and music-based datasets, it is shown that AR-VAE can work with different types of data, model architectures, and a wide range of data attributes. AR-VAE also leads to better disentanglement of the latent space and creates

meaningful attribute-based interpolations while preserving the content of the original data. This superior performance is achieved without any significant drop in the reconstruction quality of the model. While AR-VAE is designed to work with VAE-based architectures, the proposed regularization method should also work for other types of generative models such as Auto-Encoders [84], GANs [76], and flow-based models [77].

The most obvious application area for AR-VAE is in building interactive tools or plugins to aid composers and music creators. For instance, composers would be able to manipulate different attributes of the composed music to try different ideas and meet specific compositional requirements. This would allow fast iteration and would be especially useful for novices and hobbyists. Since the method is agnostic to how the attributes are computed, it can potentially be useful to manipulate high-level musical attributes such as tension and emotion. This will be particularly useful for music generation in the context of video games where the background music can be suitably changed to match the emotional context of the game and the actions of the players. The advantages of the AR-VAE framework for music generation were demonstrated in a recent study conducted by Tan and Herremans [143]. In this study to disentangle note density and rhythm density in polyphonic piano music, AR-VAE was compared against three different supervised regularization and conditioning-based methods (GLSR-VAE [67], CVAE [97], Fader Networks [100]), and was shown to outperform them in two out of the three considered metrics. In particular, AR-VAE was significantly better at the *linearity* metric which meant that attribute values obtained were directly proportional to the latent code of the regularized dimension. This ensured that AR-VAE was the most suitable to enable "*fader-like*" control during data generation.

AR-VAE can be also used in other domains as a building block to create several interesting and useful applications for context-driven data generation. In the context of images, it can be used to manipulate attributes in image or photo-based applications such as FaceApp[3] or Prisma.[4] There is also the possibility of using this for speech generation applications. For

[3]https://faceapp.com/app, last accessed: 20th July 2020
[4]https://prisma-ai.com, last accessed: 20th July 2020

instance, the ability to manipulate the prosody of the generated speech can make mobile voice assistants more realistic.

There are, however, some limitations of the approach which can open up avenues for future research. The regularization loss is currently designed to work with continuous attributes. While there is some evidence that the method can be applied to discrete categorical attributes (e.g., *shape* in the 2-d sprites dataset), additional experiments are be needed to ascertain this. Moreover, the current formulation is not suitable for binary attributes such as the ones used in Fader networks [100]. It is also observed that the choice of attributes seems to play an important role in the training process. While it is possible to jointly regularize multiple attributes in most cases (as seen in Sect. 4.3.4 and 4.3.5), strongly correlated attributes can lead to latent spaces which are not interpretable with respect to every single attribute. This results in poor control over some attributes and reduced content preservation and coherence in the interpolations. While this is a limitation, independent control over strongly correlated attributes is probably not a necessary requirement for a useful generative model. Even in cases where such control is desired, two or more separately trained AR-VAE models, which regularize those attributes individually, can be used instead.

CHAPTER 5
LATENT SPACE TRANSFORMATION

Chapter 3 of this thesis presented a learning method to leverage high-level information in entangled latent spaces, Chapter 4 presented a regularization method to learn disentangled latent spaces such that specific dimensions end up encoding selected data attributes. This chapter explores learning disentangled representations from an existing entangled latent space. This would be accomplished by learning a transformation to a secondary latent space where different attributes are regularized along its specific dimensions. This approach has some potential benefits over basic regularization-based methods:

(a) It can be built on top of existing latent spaces and hence, the reconstruction accuracy of the base model will not be affected by additional regularization terms.

(b) It can be implemented even if only part of the data has information about the attributes. This is could be useful for semi-supervised learning scenarios.

(c) Multiple transformations can be learnt from the same latent space to work for different attributes. This will also prevent the need to retrain the base model when adding new attributes.

The main motivation behind this approach stems from the Lens framework proposed by Adel et al. [105]. The key idea is to take the latent space Z of a VAE model which has been optimized for reconstruction accuracy (resulting in possibly entangled representations) and transform it into a secondary latent space Z^*. This secondary/transformed latent space can learn a disentangled representation by conditioning it on an attribute space A comprising of several attributes of the data. A schematic of the lens model framework is shown in Figure 5.1. The framework is trained using a VAE-style procedure — an inference model is

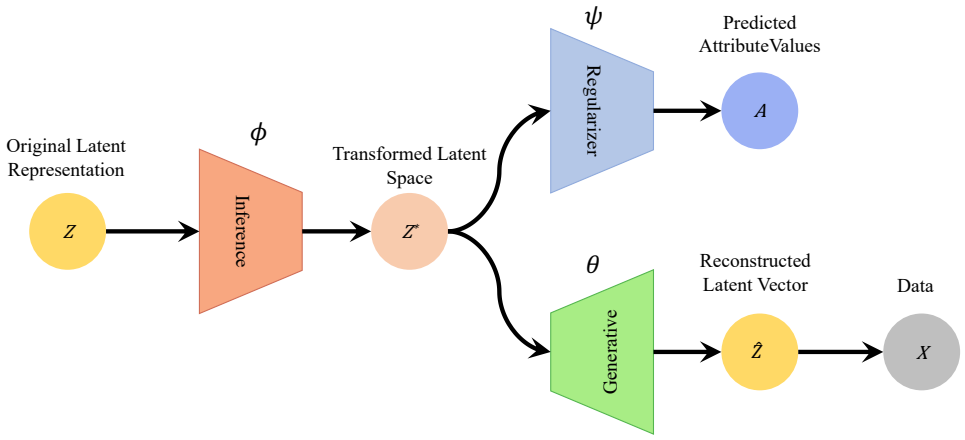


Figure 5.1: Schematic of the *Lens* framework. Z is the latent representation of a VAE trained to maximize reconstruction accuracy over the data-space X. Z^* is the transformed latent space which has been optimized for interpretability with respect to an attribute space A. The transformation is learned using an inference model parametrized by ϕ. The generative model mapping to $\hat{Z}$ parametrized by θ. The regularizer predicting the attributes A is parametrized by ψ.

used to map the original latent vectors $\mathbf{z} \in Z$ to vectors $\mathbf{z}^* \in Z^*$, while a generative model is used to reconstruct the corresponding latent vector $\mathbf{z}$ in the original latent space. In practice, the inference and generative models are implemented using a series of non-linear invertible transformations called *normalizing flows* [77, 144], which can be used to construct complex posterior distributions and are parametrized by ϕ, θ. In addition, the transformed latent space is regularized to be meaningful with respect to the attributes by using a regularizer parametrized by ψ. In the original Lens framework, this regularizer was designed for categorical attributes and was implemented using simple linear classifiers attached to one (or more) dimensions of Z^* to predict the different attribute values.

While the Lens framework has been applied to image-based data using categorical attributes, its performance has neither been investigated for music data nor for continuous attributes. In this chapter, I investigate the use of the AR-VAE regularization technique in tandem with the Lens framework to obtain a transformed latent space which is easily controllable with respect to continuous musical attributes. Details of this method are described in Section 5.1. The experimental set-up is discussed in Section 5.2 followed by

the results in Section 5.3. Finally, Section 5.4 summarizes the main findings in this chapter and outlines avenues for future research.

5.1 Method

The proposed method (see schematic shown in Figure 5.1) combines the Lens framework with the AR-VAE regularization. It can be broken down into two components:

(a) Learning the mapping between Z and Z^*.

(b) Regularizing Z^* to improve controllability with respect to different attributes.

The combined model is referred to as Lens Attribute-Regularized VAE (LAR-VAE). The task of learning a mapping between the two latent spaces (Z and Z^*) can be interpreted as learning a transformation between one probability distribution to another. This is accomplished by using normalizing flows [145, 77]. A brief background on normalizing flows is presented first.

5.1.1 Background on Normalizing Flows

A normalizing flow is a sequence of invertible transformations that is used to map one probability distribution to another. Each individual transformation is an invertible, smooth function $f : \mathbb{R}^{\mathbb{D}} \rightarrow \mathbb{R}^{\mathbb{D}}$ with an inverse $g = f^{-1}$, such that $g \circ f(\mathbf{x}) = \mathbf{x}$. If a random variable $\mathbf{z}$ with a distribution $p(\mathbf{z})$ is transformed to another random variable $\mathbf{z}'$ using f, such that $\mathbf{z}' = f(\mathbf{z})$, the distribution of the transformed random variable becomes (using the chain rule and properties of Jacobians of invertible functions):

$$p(\mathbf{z}') = p(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}'} \right| = p(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \tag{5.1}$$

The log-probability thus becomes:

$$\log p(\mathbf{z}') = \log p(\mathbf{z}) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}} \right| \tag{5.2}$$

By combining a sequence of T such transformations, arbitrarily complex densities can be constructed:

$$\mathbf{z}^* = \mathbf{z}_T = f_T \circ f_{T-1} \circ \cdots \circ f_2 \circ f_1(\mathbf{z}) \tag{5.3}$$

The probability density $p(\mathbf{z}^*|\mathbf{z})$ can be written (using Equation 5.2) as:

$$p(\mathbf{z}^*|\mathbf{z}) = \log p_T(\mathbf{z}_T|\mathbf{z}) = \log p(\mathbf{z}_0|\mathbf{z}) - \sum_{t=1}^{T} \log \left| \det \frac{\partial f_t}{\partial \mathbf{z}_{t-1}} \right| \tag{5.4}$$

where, $\mathbf{z}_0$ is an initial random variable with density $p_0(\mathbf{z}_0)$
The individual transformations f_t (referred to as flows) are chosen such that they are easily invertible and have an easy to compute Jacobian. Further, the invertible nature of the mapping allows estimating $\mathbf{z}$ from $\mathbf{z}^*$.

$$\mathbf{z} = f_0^{-1} \circ f_1^{-1} \circ \cdots \circ f_{T-1}^{-1} \circ f_T^{-1}(\mathbf{z}^*) \tag{5.5}$$

There are a number of different models which use neural-networks to construct these flows, e.g., planar flows [77], NICE (Non-Linear Independent Component Estimation) [146], RealNVP (Real-valued Non-Volume Preserving) [147]. For this study, the Glow model, proposed by Kingma and Dhariwal [148], is used. This model extends the NICE and RealNVP models by using invertible 1×1 convolutions paired with a multi-scale architecture [147], and was able to show significant improvement in density estimation tasks across several image-based datasets [148].

5.1.2 Lens Model Training

Without taking any attribute specific information into account and using principles of variational inference, the marginal likelihood of a data-point $\mathbf{z}$ for the Lens model is given by:

$$
\begin{aligned}
\log p(\mathbf{z}) &\geq \log \int_{\mathbf{z}^*} p(\mathbf{z}, \mathbf{z}^*) d\mathbf{z}^* \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z}^*|\mathbf{z})}[p(\mathbf{z}^*) p_\theta(\mathbf{z}|\mathbf{z}^*) / q_\phi(\mathbf{z}^*|\mathbf{z})] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}^*|\mathbf{z})}[\log p(\mathbf{z}^*) + \log p_\theta(\mathbf{z}|\mathbf{z}^*) - \log q_\phi(\mathbf{z}^*|\mathbf{z})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}^*|\mathbf{z})}[\log p_\theta(\mathbf{z}|\mathbf{z}^*) + \log p(\mathbf{z}^*) - \log q(\mathbf{z}_0|\mathbf{z}) + \sum_{t=1}^{T} \log \left| \det \frac{\partial f_t}{\partial \mathbf{z}_{t-1}} \right|]
\end{aligned}
$$

where the last equality is obtained by using Equation 5.4 and rearranging the terms. The loss function for the Lens model L_{Lens} which seeks to minimize the negative log-probability of $p(\mathbf{z})$ is defined as:

$$
L_{\text{Lens}} = -\mathbb{E}_{q_\phi(\mathbf{z}^*|\mathbf{z})}[\log p_\theta(\mathbf{z}|\mathbf{z}^*) + \log p(\mathbf{z}^*) - \log q(\mathbf{z}_0|\mathbf{z}) + \sum_{t=1}^{T} \log \left| \det \frac{\partial f_t}{\partial \mathbf{z}_{t-1}} \right|] \quad (5.6)
$$

In order to regularize the set of $\mathbb{L}$ attributes along the different dimensions, additional regularizers are used based on the attribute-based regularization loss from Equation 4.3. Thus, the overall loss for the LAR-VAE model is formulated as:

$$
L_{\text{LAR-VAE}} = L_{\text{Lens}} + \gamma \sum_{l=0}^{\mathbb{L}-1} L_{r_l, a_l} \quad (5.7)
$$

where L_{r_l,a_l} is the regularization loss for the attribute a_l with r_l as the index of the regularized dimension in the transformed latent space, and γ is a tunable hyperparameter which is referred to as the regularization strength. Note that the regularization loss L_{r_l,a_l} is computed on using the transformed latent vector $\mathbf{z}^*$. The overall learning algorithm for LAR-VAE is shown in Algorithm 2.

Algorithm 2: Learning algorithm for LAR-VAE

Input: observations and attribute labels $(\mathbf{x}_i, \{a_l\}_i)_{i=1}^N$ (N is the number of examples in the dataset, $l \in [0, \mathbb{L})$ where $\mathbb{L}$ is the number of attributes), batch-size m, indices of the latent dimensions to be regularized $\{r_l\}_{l=0}^{\mathbb{L}-1}$, pre-trained base-VAE (vanilla-VAE) encoder and decoder parameters Φ, Θ, initialized lens model parameters: ϕ, θ, neural network optimizer g

repeat

  Randomly sample a batch of m data-points $(\mathbf{x}_i, \{a_l\}_i)$

  Compute the latent representations $\mathbf{z}_i \sim p_\Phi(\mathbf{z}_i|\mathbf{x}_i)$

  Compute $\mathbf{z}^*$ and $L_{\text{Lens}}(\theta, \phi)$ using Equation 5.6

  for $l \in [0, \mathbb{L})$ **do**

    Compute L_{r_l, a_l} on $\mathbf{z}^*$ using Equation 4.3

  end

  Compute $L_{\text{LAR}-\text{VAE}}(\theta, \phi)$ using Equation 5.7

  Update LAR-VAE model parameters: $\theta, \phi \leftarrow g(L_{\text{LAR}-\text{VAE}}(\theta, \phi))$

until *convergence of objective*

5.2 Experimental Setup

For evaluating the performance of the LAR-VAE model, the same overall experimental set-up is used as in Section 4.2. All the four datasets (dSprites, Morpho-MNIST, Bach Chorales, and Folk Music) and their respective attributes from Section 4.2.1 are considered. For each dataset, vanilla-VAEs are trained to obtain the base latent spaces Z. LAR-VAE models are trained on top of these vanilla-VAEs. The performance of the LAR-VAE models is compared against corresponding β-VAE and AR-VAE models from Chapter 4 which were trained directly on the raw data.

5.2.1 Model Architectures

For the vanilla-VAEs, the same architectures as those used in Section 4.2.3 (details provided in Appendix A.2) are used. Convolutional architectures are used for image-based datasets whereas recurrent architectures are used with the music-based datasets.

For the LAR-VAE model, the architecture details of the Glow-based Lens model are provided in Appendix B.1.

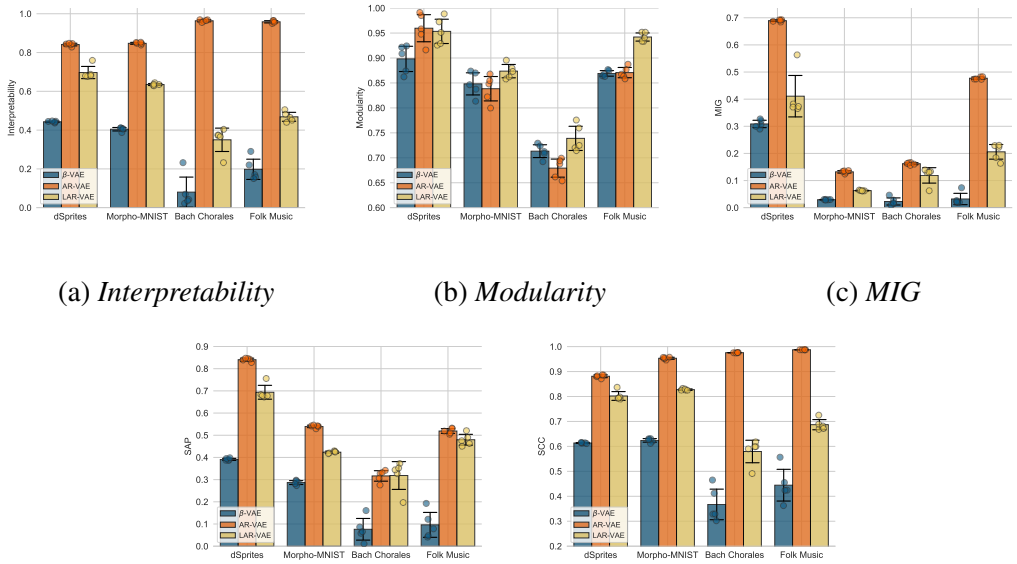
5.2.2 Implementation Details

Consistent with the previous chapter, each dataset is divided into a train-validation-test split using an 80%-15%-5% ratio. The train and validation sets are used to train the models and tune hyperparameters. The held-out test sets are used for evaluation.

For Vanilla-VAE training, $\beta = 1.0$ is used for the image-based datasets and $\beta = 1\text{e}{-3}$ is used for the music-based datasets. Thus, effectively, for music-based datasets, the vanilla-VAEs yield the same results as β-VAEs. For LAR-VAE, the models for the image datasets are trained with $\gamma = 10.0$, whereas those for the music datasets are trained with $\gamma = 1.0$. For each dataset, both the vanilla-VAE and the corresponding LAR-VAE models are trained with 5 different random initializations. All models for the same dataset are trained for the same number of epochs (models for both image-based datasets, Folk Music, and Bach Chorales are trained for 100, 15, and 30 epochs, respectively). Batch-size is fixed at 16 and optimization is carried out using the ADAM optimizer [132] with a fixed learning rate of $1\text{e}{-4}$ (other optimization parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1\text{e}{-8}$).

5.3 Results and Discussion

The performance of the LAR-VAE model is evaluated using different experiments. First, the disentanglement performance and reconstruction accuracy are presented. These experiments follow the same protocol as in Section 4.3. Based on the results, a couple of additional experiments are conducted which provide deeper insights into how the correlation between the attributes as well as the information loss in the entangled vanilla-VAE latent space affects the performance of the LAR-VAE model. Finally, a discussion on the interpolation capabilities and limitations of the proposed model is presented.



(a) *Interpretability* (b) *Modularity* (c) *MIG*

(d) *SAP* (e) *SCC*

Figure 5.2: Bar plots for disentanglement performance (higher is better) for LAR-VAE model across different datasets compared against β-VAE and AR-VAE. The circular dots denote results for each random seed. Error bars correspond to one standard deviation.

5.3.1 Disentanglement

The disentanglement performance based on different metrics for all the datasets is shown in Figure 5.2. The LAR-VAE model performs better than the β-VAE models across all metrics. This indicates that, compared to the vanilla-VAE latent space, the learnt transformation leads to better disentanglement in the Z^* latent space of LAR-VAE.

However, for most datasets and metrics, AR-VAE outperforms the LAR-VAE model. Note that the AR-VAE models are trained on the raw data as opposed to LAR-VAE which is trained on top of the latent space of a vanilla-VAE. This suggests that either the LAR-VAE training method is not able to disentangle different attributes from the original latent space, or information pertaining to the different data attributes might already be lost during the vanilla-VAE training process. This is investigated later in Section 5.3.4.

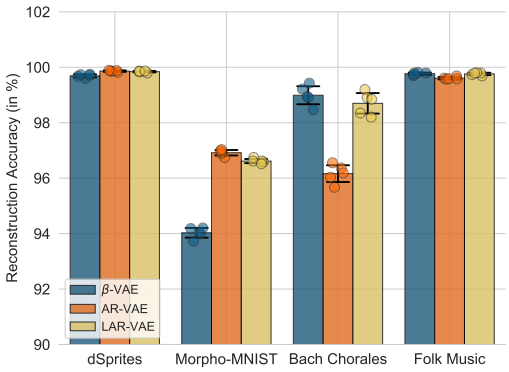


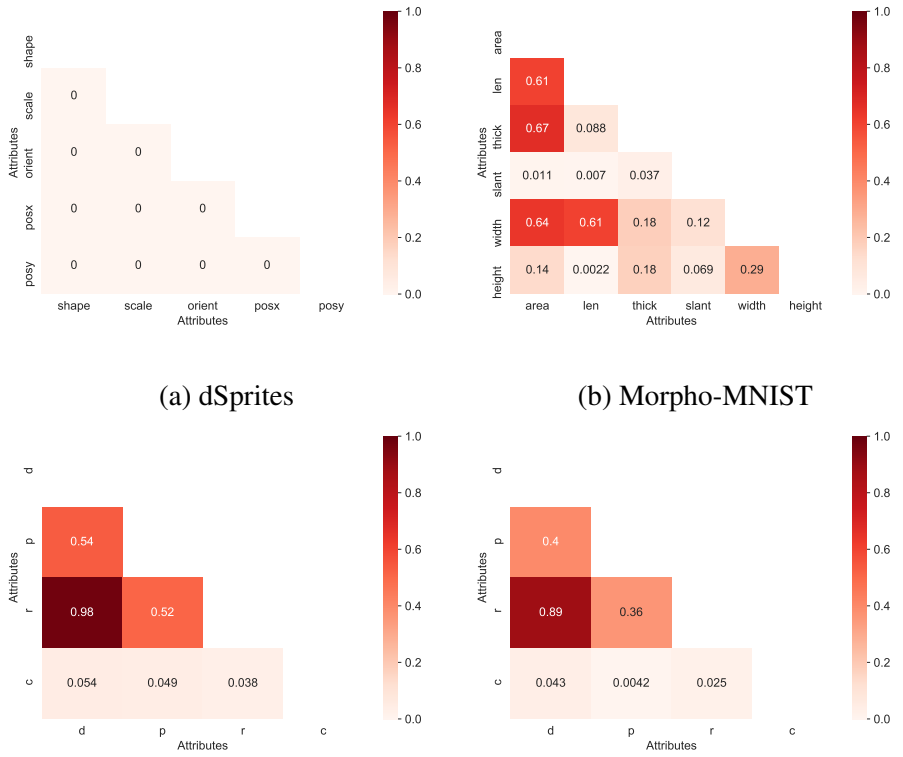
Figure 5.3: Reconstruction results (higher is better) for LAR-VAE compared to other models across the different datasets.

5.3.2 Reconstruction Accuracy

The LAR-VAE training process does not interfere with the training of the vanilla-VAE. In addition, the mapping between the vanilla latent space and the transformed latent space is invertible. Consequently, the reconstruction accuracy of the model is not affected which is clear from the results shown in Figure 5.3. Note that for Morpho-MNIST dataset, the β-VAE performs worse as it uses $\beta = 4.0$ whereas the LAR-VAE is trained on top of a vanilla-VAE with $\beta = 1.0$ The LAR-VAE model has similar reconstruction performance as the other two models for three out of the four datasets. However, for the smaller-sized Bach Chorales dataset, it performs better than AR-VAE which has to not only minimize the reconstruction loss but also simultaneously minimize the different attribute regularization losses.

5.3.3 Attribute Correlations

A closer inspection of the Z^* latent spaces revealed that, for LAR-VAE models trained on music datasets, often, two attributes end up being encoded along the same dimension of the latent space. For example, *note density* and *rhythmic complexity* are encoded along the same dimensions. Similarly, *pitch range* and *contour* are encoded along the same



(a) dSprites

(b) Morpho-MNIST

(c) Bach Chorales

(d) Folk Music

Figure 5.4: Attribute correlations for the different datasets. Absolute values of the Spearman correlation coefficient are shown. Darker colors (higher values) indicate a stronger correlation between the pair of attributes defined by row and column. For (c) and (d) d: Note Density, p: Pitch Range, r: Rhythmic Complexity, c: Contour.

dimension. This is in spite of the attribute regularization loss which tries to force encoding along different dimensions. However, the same is not observed for the image-based datasets. A possible reason for this could be the high degree of correlation between certain attributes. Some experiments in the previous chapter (see Section 4.3.3 and 4.3.5) also suggested that correlations between attributes might play a role in the performance of the AR-VAE regularization.

Figure 5.4 shows the Spearman's coefficient of correlation for different pairs of attributes across all the datasets. Overall, the degree of correlation between the attributes is lower in the image datasets compared to the music datasets. The dSprites dataset has zero correlation on account of its orthogonal attributes. There is some degree of correlation between some

of the attributes in the Morpho-MNIST dataset. For instance, *area* is correlated with both *thickness* and *length*. Also, *length* is correlated with *width*.

In contrast, compare this is to the music datasets where there is a high correlation between *note density* and *rhythmic complexity*. This explains why these two attributes end up entangled along the same dimension in the transformed latent space of the LAR-VAE model. For AR-VAE, even though there is some degree of disentanglement, this causes problems during data generation. Traversals along the dimension regularizing one attribute tend to cause a change in the other and vice versa (see Section 4.3.3).

The correlation plots in Figure 5.4 still do not explain why *pitch range* and *contour* end up being entangled for LAR-VAE. This is explained by taking a closer look at the *contour* attribute. Unlike the other three attributes, *contour* is symmetric about zero. Consequently, its absolute value is strongly correlated with *pitch range* (0.74 for the Bach Chorales dataset, 0.54 for the Folk Music dataset) which might result in the entanglement between these two attributes.

5.3.4 Information Loss in Entangled Latent Spaces

It is observed in Section 5.3.1 that the LAR-VAE model is not able to disentangle different attributes as effectively as the AR-VAE model. One of the possible reasons for this could be that the vanilla-VAE latent space does not contain enough information about these attributes for the LAR-VAE training process to learn from. This experiment investigates this by measuring the predictive power of the vanilla-VAE latent representation to predict the different attribute values as a proxy to the information content. This is accomplished by fitting a linear regression model to predict a given attribute from the latent representations corresponding to data-points from the held-out test set. The final metric is computed by averaging the R^2 score across all attributes. Figure 5.5 shows these results along with those for the AR-VAE latent space.

The results indicate that, compared to the AR-VAE model, the attributes might only be

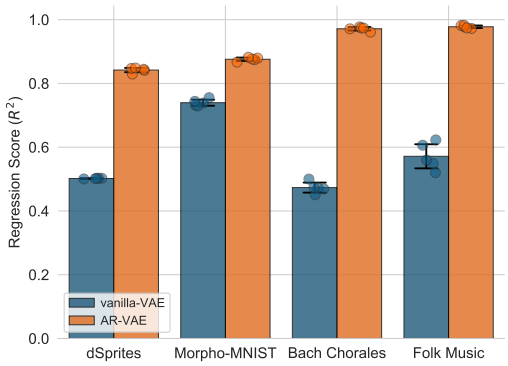
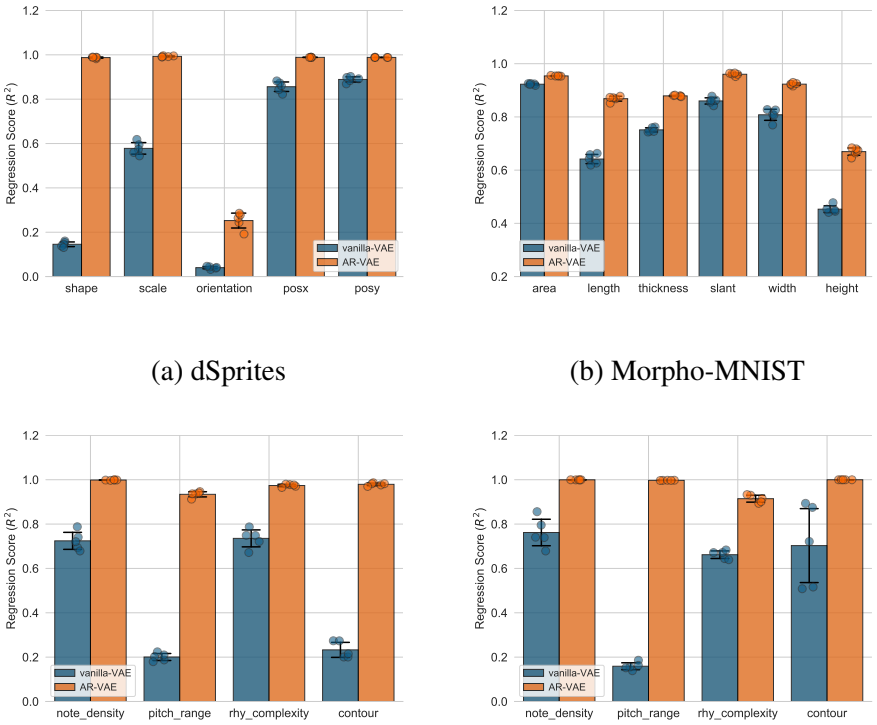


Figure 5.5: Mean regression score (R^2, higher is better) for attribute prediction from latent codes. Numbers are averaged across all attributes for a dataset.

partially encoded in the vanilla-VAE latent space. This is reflected in the lower mean R^2 scores for the vanilla-VAE models. This holds true across all the datasets. The difference seems to be larger in the case of music datasets. Consequently, the LAR-VAE models, which are trained on top of the vanilla-VAE latent space do not have enough information to use in order to disentangle the attributes in a meaningful manner. Another possible interpretation of these results could be that the attributes are entangled to such a large extent that disentanglement is difficult.

There are also considerable differences between individual attributes across datasets which can be seen in Figure 5.6. For instance, *pitch range* has the lowest R^2 score across both music datasets. The results for the dSprites dataset are particularly interesting. dSprites is an artificial dataset with orthogonal attributes. The high reconstruction accuracy of the vanilla-VAE would indicate that there is no loss of information with respect to any attribute in the latent representation. However, the R^2 score for the *pos-x*, *pos-y*, and *scale* attributes is much higher than the *shape* and *orientation* attributes. This suggests that the latter two attributes are entangled to a greater degree than the former three. This behavior is also reflected in the interpolation results shown in Figure 4.5 for the β-VAE model which, in the absence of any explicit regularization, seems to struggle with the *shape* and *orientation* attributes. Thus, for unsupervised methods, the choice of attributes might play a crucial role



(a) dSprites

(b) Morpho-MNIST

(c) Bach Chorales

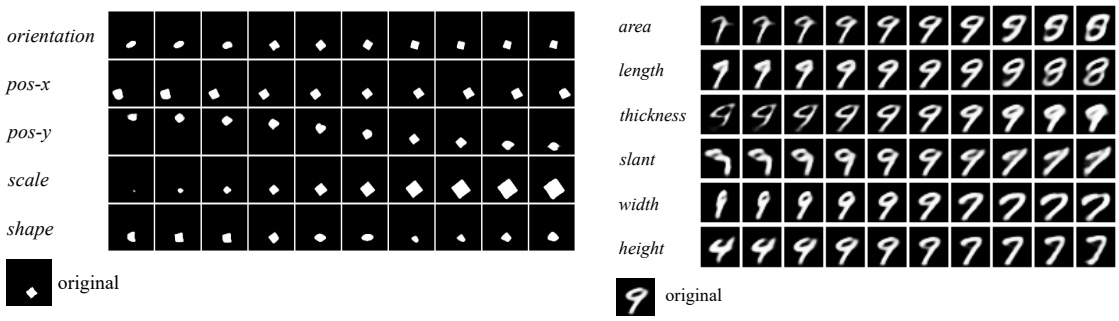
(d) Folk Music

Figure 5.6: Mean regression score (R^2, higher is better) for attribute prediction from latent codes. Results for individual attributes across the datasets are shown.

in how effectively they are disentangled in the learnt representations.

5.3.5 Discussion

Overall, the LAR-VAE model seems to work for image datasets but struggles for the music datasets. This is also reflected in the generated interpolations by traversing along the transformed latent space of the LAR-VAE model. Figure 5.7 shows some examples for the dSprites and Morpho-MNIST datasets. While these results are not as good as those obtained for AR-VAE (compare with Figures 4.5 and 4.6), the results are somewhat better than β-VAE. This is in line with the objective disentanglement results obtained in Section 5.3.1. It can also be seen that attributes which have a lower R^2 score in Figure 5.6 (*shape* and *orientation* for dSprites, *length*, and *height* for Morpho-MNIST) have a relatively poorer



(a) dSprites

(b) Morpho-MNIST

Figure 5.7: Interpolation results using LAR-VAE model for the image-based datasets. For each row different images are generated by changing the latent code for the corresponding regularized dimension in the transformed latent space of the LAR-VAE model.

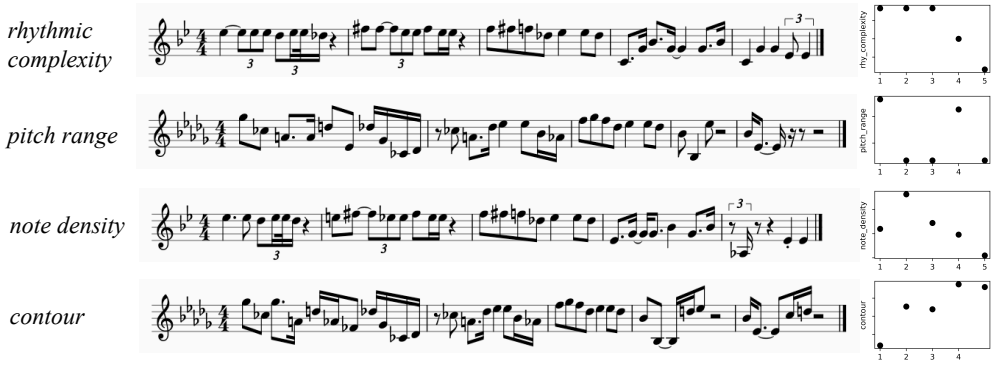


Figure 5.8: Interpolations results using LAR-VAE model for the Folk Music dataset. In each row the different measures are generated by changing the latent code for the corresponding regularized dimension in the transformed latent space of the LAR-VAE model. The plots on the right show how the attribute values change with the increase in the latent code.

quality of interpolations as well. This seems to confirm the hypothesis that certain attributes are harder to disentangle.

Contrast this to the interpolation results for the Folk Music dataset which are shown in Figure 5.8. There is no visible control over any of the attribute values. The generated measures are not coherent in any manner. The AR-VAE model, in contrast, performs considerably better (compare with Figure 4.7(b)).

Thus, while using the LAR-VAE model can lead to some objective improvement over β-VAE in terms of disentangling different attributes, the resulting latent spaces are still not

very useful from the perspective of controllable music generation. This inferior performance (compared to image datasets) might be due to several reasons. The music datasets used for this study represent real-world music while the image datasets are considerably simpler in nature. On the other hand, the chosen attributes for the music datasets might also not be amenable to effective disentanglement. For instance, while *rhythmic complexity* might be an interesting property to control from a music generation perspective, it might not be an *inherent* factor of variation in music data for unsupervised methods (such as β-VAE and vanilla-VAE) to disentangle. These results indicate that using supervised methods to disentangle different attributes of interest might be the way forward for improving controllability in music generation models.

5.4 Conclusion

This chapter investigated the problem of learning disentangled latent representations from entangled representations. To achieve this, a normalizing flow-based model is used to learn a non-linear invertible mapping between the latent representation of a vanilla-VAE and a new transformed latent space. This is combined with the AR-VAE regularization proposed in Chapter 4 to force the transformed latent space dimensions to selectively encode specific attributes.

The resulting LAR-VAE model shows better disentanglement than the β-VAE model without sacrificing any reconstruction quality. However, the disentanglement performance is inferior to that of the AR-VAE model which learns representations directly from the raw data. Further investigation using a couple of experiments revealed that the transformed latent space is not very useful from the perspective of controllable generation on account of two factors.

First, attributes which are strongly correlated are entangled along the same dimension in the transformed latent space. This problem is especially prevalent in music datasets where certain pairs of chosen attributes are correlated either directly (*note density* with *rhythmic*

complexity) or indirectly (*pitch range* with *contour*). This highlights the importance of carefully choosing attributes while designing models for controllable music generation.

Second, latent spaces trained without any explicit supervision, e.g., those learnt by vanilla or β-VAE, might either lose information (specific to certain attributes of interest) or might learn highly entangled representations which are hard to disentangle later. Specifically, regularization-based disentanglement methods which try to force a single dimension of the latent space to encode these attributes (e.g., attribute-based regularization used here, GLSR-VAE [72]) might not work very well for these attributes. A possible way to tackle this could be to first use a subset of the latent code to localize these attributes [96, 111]. Control over these attributes can then be provided by using specific single-dimension regularizers on these sub-latent spaces [143].

CHAPTER 6

DISENTANGLEMENT LEARNING FOR MUSIC

Learning compact and *disentangled* representations (see Figure 6.1 for an illustration) from given data, where important factors of variation are clearly separated, is considered useful for generative modeling and for improving performance on downstream tasks (such as speech recognition, speech synthesis, vision, and language generation [149, 150, 151]). The importance of learning disentangled representations has also been shown in Chapters 4 and 5 of this thesis.

As discussed in Section 2.2, there exists a large body of research in the machine learning community focused on developing algorithms for learning disentangled representations. These span unsupervised [86, 92, 93, 114], semi-supervised [81, 152, 102] and supervised [100, 67, 83, 96] methods. However, a vast majority of these algorithms are designed, developed, tested, and evaluated using data from the image or computer vision domain. This restricted focus on a single domain raises concerns about the generalization of these methods [89]. In addition, research on disentanglement learning for music has often been application-oriented with researchers using their own problem-specific datasets and attributes. Contrary to the image-domain, there are no standard datasets for conducting systematic studies on disentanglement learning in the context of music.

This chapter[1] attempts to address the above limitation by proposing an algorithmically generated symbolic music dataset for disentanglement learning which is presented in Section 6.2. This dataset is used to conduct a systematic study on music disentanglement spanning several unsupervised and supervised disentanglement learning algorithms which are presented in Section 6.3 and 6.4, respectively. Finally, Section 6.5 summarizes the findings and outlines directions for future research. Before diving into the dataset design

[1] Parts of this chapter have been published in [69].

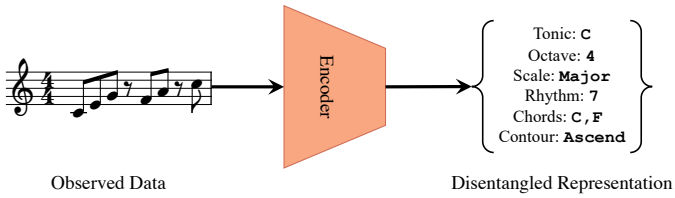


Figure 6.1: Disentanglement representation learning example where a high dimensional observed data is disentangled into a low dimensional representation comprising of semantically meaningful factors of variation.

and other details, a discussion on the need for such a dataset is presented first in Section 6.1.

6.1 Motivation

In Chapter 2, I briefly discussed the idea behind representation learning, where, given an observation $\mathbf{x}$, the task is to learn a representation $r(\mathbf{x})$ which "makes it easier to extract useful information when building classifiers or other predictors" [19]. The fundamental assumption is that any high-dimensional observation $\mathbf{x} \in X$ (where X is the data-space) can be decomposed into a semantically meaningful low dimensional latent variable $\mathbf{z} \in Z$ (where Z is the latent space). Given a large number of observations in X, the task of disentanglement learning is to estimate this low dimensional latent space Z by separating out the distinct factors of variation [19]. An ideal disentanglement method ensures that changes to a single underlying factor of variation in the data modify only a single factor in its representation [89]. From a generative modeling perspective, it is also important to learn the mapping from Z to X to enable better control over the generative process.

6.1.1 Diversity in Disentanglement Learning

Most state-of-the-art methods for unsupervised disentanglement learning are based on the VAE [75] framework. The key idea behind these methods is that factorizing the latent representation to have an aggregated posterior should lead to better disentanglement [89]. This is achieved using different means, e.g., imposing constraints on the information capacity

of the latent space [86, 90, 91], maximizing the mutual information between a subset of the latent code and the observations [82], and maximizing the independence between the latent variables [92, 93]. However, unsupervised methods for disentanglement learning are sensitive to inductive biases (such network architectures, hyperparameters, and random seeds) and consequently, there is a need to properly evaluate such methods by using datasets from diverse domains [89].

Apart from unsupervised methods for disentanglement learning, there has also been some research on semi-supervised [152, 102] and supervised [83, 100, 153, 66] learning techniques to manipulate specific attributes in the context of generative models. In these paradigms, a supervised loss based on attribute labels is used in addition to the unsupervised loss. Available labels can be utilized in various ways. They can help with disentangling known factors (e.g., digit class in MNIST) from latent factors (e.g., handwriting style) [101], or supervising specific latent dimensions to map to specific attributes [67]. However, most of these approaches are evaluated using image domain datasets.

Tremendous interest from the machine learning community has led to the creation of benchmarking datasets (albeit image-based) specifically targeted towards disentanglement learning such as dSprites [64], 3D-Shapes [154], 3D-chairs [155], MPI3D [156], most of which are artificially generated and have simple factors of variation. While one can argue that artificial datasets do not reflect real-world scenarios, the relative simplicity of these datasets is often desirable since they enable rapid prototyping. In addition, most of these datasets have independent or orthogonal factors of variation which ensure consistent and systematic evaluation.

6.1.2 Consistency in Music-based Studies

Representation learning has also been explored in the field of MIR. Much like images, learning better representations has been shown to work well for MIR tasks such as composer classification [157, 158], music tagging [159], and audio-to-score alignment [160]. The

idea of disentanglement has been particularly gaining traction in the context of interactive music generation models [66, 111, 112, 71]. Disentangling semantically meaningful factors can significantly improve the usefulness of music generation tools. Many researchers have independently tried to tackle the problem of disentanglement in the context of symbolic music by using different musically meaningful attributes such as genre [111], note density [67], rhythm [112], and timbre [161]. Some of these studies were already discussed in Section 2.3. However, these methods and techniques have all been evaluated using different datasets which makes a direct comparison difficult. Part of the reason behind this lack of consistency is the difference in the problems that these methods were looking to address. However, the availability of a common dataset allowing researchers to easily compare algorithms and test their hypotheses will surely aid systematic research.

6.2 dMelodies Dataset

In this section, the dMelodies dataset is introduced which is specifically designed to facilitate systematic studies in music disentanglement. Before diving deeper into the methodology used to create the dataset, the adopted design choices are presented first.

6.2.1 Design Principles

To enable objective evaluation of disentanglement algorithms, one needs to either know the ground-truth values of the underlying factors of variation for each data point or be able to synthesize the data points based on the attribute values. The dSprites dataset [64], for instance, consists of single images of different 2-dimensional shapes with simple attributes specifying the position, scale, and orientation of these shapes against a black background. The design of dMelodies is loosely based on the dSprites dataset. The following principles were used to finalize other design choices:

(a) The dataset should have a simple construction with homogeneous data points and intuitive factors of variation. It should allow for easy differentiation between data

points and have clearly distinguishable latent factors.

(b) The factors of variation (attributes) should be orthogonal, i.e., changing any one factor should not cause changes to other factors. While this is not always true for real-world data, it enables consistent objective evaluation.

(c) There should be a clear one-to-one mapping between the latent factors and the individual data points. In other words, each unique combination of the factors should result in a unique data point.

(d) The factors of variation should be diverse. In addition, it would be ideal to have the factors that span different types such as discrete, ordinal, categorical, and binary.

(e) Finally, the different combinations of factors should result in a dataset large enough to train deep neural networks. Based on the size of the different image-based datasets [64, 162], this should be of the order of at least a few hundred thousand data points.

6.2.2 Dataset Construction

Considering the design principles outlined above, the focus of this work is on monophonic pitch sequences. While there are other options such as polyphonic or multi-instrumental music, the choice of monophonic melodies is to ensure simplicity. Monophonic melodies are a simple form of music uniquely defined by the pitch and duration of their note sequences. The pitches are typically based on the key or scale in which the melody is being played and the rhythm is defined by the onset positions of the notes.

Since the set of all possible monophonic melodies is very large and heterogeneous, the following additional constraints are imposed on the melody in order to enforce homogeneity and satisfy the other design principles:

(a) Each melody is based on a scale selected from a finite set of allowed scales. This choice of scale also serves as one of the factors of variation. The melody will also be uniquely defined by the pitch class of the tonic (root pitch) and the octave number.

(b) In order to constrain the space of all possible pitch patterns within a scale, each melody is restricted to be an arpeggio over the standard I-IV-V-I cadence chord pattern. Consequently, each melody consists of 12 notes (3 notes for each of the 4 chords).

(c) In order to vary the pitch patterns, the direction of arpeggiation of each chord, i.e., up or down, is used as a latent factor. This choice adds a few binary factors of variation to the dataset.

(d) The melodies are fixed to 2-bar sequences with the 8th note as the minimum note duration. This makes the dataset uniform in terms of sequence lengths of the data points and also helps reduce the complexity of the sequences. 2-bar sequences have been used in other music generation studies as well [67, 11]. A tokenized data representation is used such that each melody has a sequence of length 16.

(e) If the space of all possible unique rhythms is considered, the number of options will explode to $\binom{16}{12}$ which will be significantly larger than other factors of variation. Hence, the latent factor for rhythm is broken down into 2 independent factors: rhythm for bar 1 and bar 2.

(f) The rhythm of a melody is based on the metrical onset position of the notes [142]. Consequently, rhythm is dependent on the number of notes. In order to keep rhythm independent from other factors, each bar is constrained to have 6 notes (play 2 chords) thereby obtaining $\binom{8}{6}$ options for each bar.

Based on the above design choices, the dMelodies dataset consists of 2-bar monophonic melodies with 9 factors of variations listed in Table 6.1. The factors of variation were chosen to satisfy the design principles listed in Section 6.2.1. For instance, while melodic transformations such as repetition, inversion, retrograde would have made more musical sense, they did not allow the creation of a large-enough dataset with independent factors of

Table 6.1: List of different factors of variation for the dMelodies dataset. Since all factors of variation are independent, the total dataset contains 1,354,752 unique melodies.

Factor	**Abbreviation**	**# Options**	**Notes**
Tonic	*Tn*	12	C, C#, D, through B
Octave	*Oc*	3	Octave 4, 5 and 6
Scale	*Sc*	3	major, harmonic minor, and blues
Rhythm Bar 1	*R1*	28	$\binom{8}{6}$, based on onset locations of 6 notes
Rhythm Bar 2	*R2*	28	$\binom{8}{6}$, based on onset locations of 6 notes
Arp Chord 1	*A1*	2	up/down, for Chord 1
Arp Chord 2	*A2*	2	up/down, for Chord 2
Arp Chord 3	*A3*	2	up/down, for Chord 3
Arp Chord 4	*A4*	2	up/down, for Chord 4

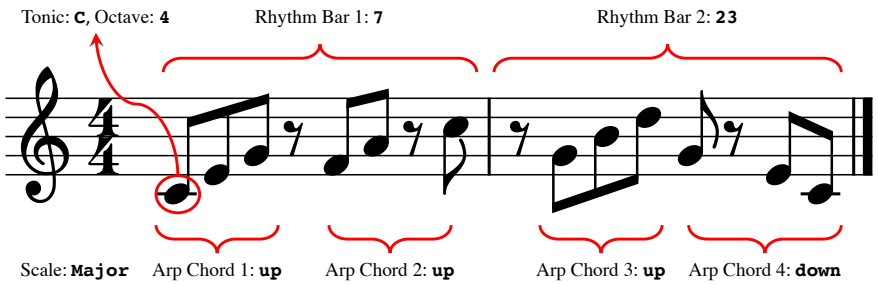


Figure 6.2: Example of a sample melody from the dMelodies dataset. Also shown are the values of the different latent factors. For rhythm latent factors, the shown value corresponds to the index from the rhythm dictionary.

variation. The resulting dataset, thus, contains simple melodies which do not adequately reflect real-world musical data. A side-effect of this choice of factors is that some of them (such as arpeggiation direction and rhythm) affect only a specific part of the data. Since each unique combination of these factors results in a unique data point we get 1,354,752 unique melodies. Figure 6.2 shows one such melody from the dataset and its corresponding latent factors. The dataset[2] is generated using the *music21* [130] python package.

[2]Released publicly at: https://github.com/ashispati/dmelodies_dataset

6.3 Experiments using Unsupervised Methods

In this section, benchmarking experiments are presented to demonstrate the performance of some of the existing unsupervised disentanglement algorithms on the dMelodies dataset and contrast the results with those obtained on the image-based dSprites dataset.

6.3.1 Experimental Setup

A recent study by Locatello et al. [89] has shown that different unsupervised disentanglement learning methods perform comparably on several image-based benchmarking datasets. To ascertain this for the dMelodies dataset, three different unsupervised disentanglement learning methods are considered: β-VAE [86], Annealed-VAE [90], and FactorVAE [93]. While all these methods strive to force the latent representation to have a factorized aggregated posterior, they use different methods to do so. The β-VAE and Annealed-VAE methods both rely on imposing constraints on the capacity of the latent bottleneck to force the encoder to learn a disentangled representation. The FactorVAE method uses an adversarial scheme to force the latent dimensions to be independent of each other.

Data Representation: A tokenized data representation similar to Section 3.3.3 is used with the 8th-note as the smallest note duration. Considering that there are no triplets in the dMelodies dataset, a uniform subdivision scheme is used. Each 8th note position is encoded with a token corresponding to the note name which starts on that position. A special continuation symbol ('__') is used which denotes that the previous note is held. A special token is used for rest. Thus, each melody is a sequence of 16 tokens.

Model Architectures: Two different VAE architectures are chosen to conduct these experiments. The first architecture, referred to as dMelodies-CNN, is based on Convolutional Neural Networks (CNNs). The second architecture, referred to as dMelodies-RNN, uses a hierarchical recurrent model [11]. For the dSprites dataset, a CNN based architecture is used which is referred to as dSprites-CNN. Details of the model architectures are provided

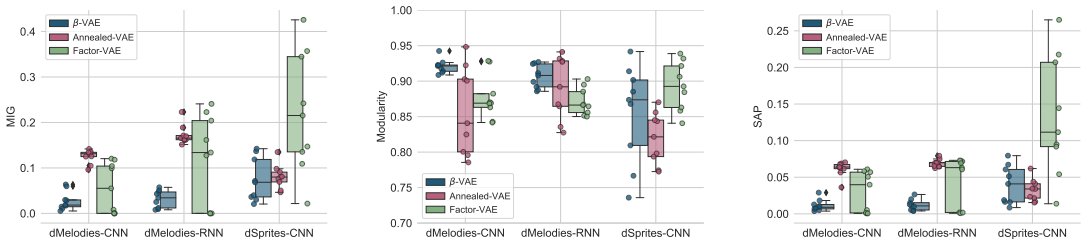
in Appendix C.1.

Hyperparameters: Each learning method has its own regularizing hyperparameter. For β-VAE, three different values of $\beta \in \{0.2, 1.0, 4.0\}$ are used. This choice is loosely based on the notion of normalized-β [86]. In addition, the KL-regularization is forced only when the KL-divergence exceeds a fixed threshold $\tau = 50$ [144, 11]. For Annealed-VAE, $\gamma = 1.0$ is fixed and three different values of capacity, $C \in \{25.0, 50.0, 75.0\}$ are used. For FactorVAE, the Annealed-VAE loss function with a fixed capacity ($C = 50$), and three different values for $\gamma \in \{1, 10, 50\}$ are chosen.

Training Specifications: For each of the above methods, model, and hyperparameter combination, 3 models with different random seeds are trained. To ensure consistency across training, all models are trained with a batch-size of 512 for 100 epochs. The ADAM optimizer [132] is used with a fixed learning rate of $1e{-}4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e{-}8$. For β-VAE and Annealed-VAE, 10 warm-up epochs are used where $\beta = 0.0$. After the warm-up, the regularization hyperparameter (β for β-VAE and C for Annealed-VAE) is annealed exponentially from 0.0 to their target values over 100000 training steps. For FactorVAE, the original implementation is used in the loss function without any annealing. The VAE optimizer is the same as mentioned earlier. The FactorVAE discriminator is optimized using ADAM with a fixed learning rate of $1e{-}4$, $\beta_1 = 0.8$, $\beta_2 = 0.9$, and $\epsilon = 1e{-}8$. Utilizing the original hyperparameters [93] for this optimizer led to unstable training on dMelodies.

For comparison with dSprites, the results for all the three methods using the dSprites-CNN model are presented. The set of hyperparameters and other training configurations are kept the same for the dSprites dataset, except for the FactorVAE where the originally proposed loss function and discriminator optimizer hyperparameters are used, as the model does not converge otherwise.

Disentanglement Metrics: For evaluating disentanglement, *MIG*, *Modularity*, and *SAP* score are used. More information about these is provided in Section 2.4. For each metric,



(a) *MIG* (b) *Modularity* (c) *SAP Score*

Figure 6.3: Overall disentanglement performance (higher is better) of different methods on the dMelodies and dSprites datasets. Individual points denote results for different hyperparameter and random seed combinations.

the mean across all latent factors is used for aggregation.

6.3.2 Disentanglement

In this experiment, the comparative disentanglement performance of the different methods on dMelodies is evaluated. The result for each method is aggregated across the different hyperparameters and random seeds. Figure 6.3 shows the results for all three disentanglement metrics. The results for the dSprites dataset are also shown for comparison. The following observations can be made:

(a) First, the performance of different methods on dMelodies is compared. Annealed-VAE shows better performance for MIG and SAP. These metrics indicate the ability of a method to ensure that each factor of variation is mapped to a single latent dimension. The performance in terms of Modularity is similar across the different methods. High Modularity indicates that each dimension of the latent space maps to only a single factor of variation. For dSprites, FactorVAE seems to be the best method overall across metrics. However, the high variance in the results shows that the choice of random seeds and hyperparameters is probably more important than the disentanglement method itself. This is in line with observations in previous studies [89].

(b) Second, there is no significant impact of model architecture on disentanglement per-

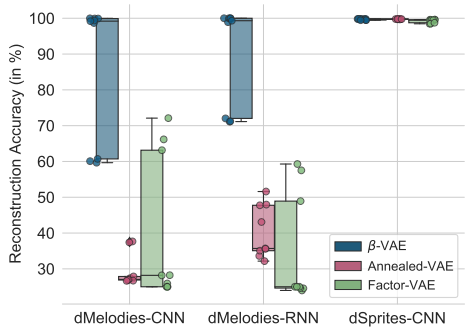


Figure 6.4: Overall reconstruction accuracies (higher is better) of the different methods on the dMelodies and dSprites datasets. Individual points denote results for different hyperparameter and random seed combinations.

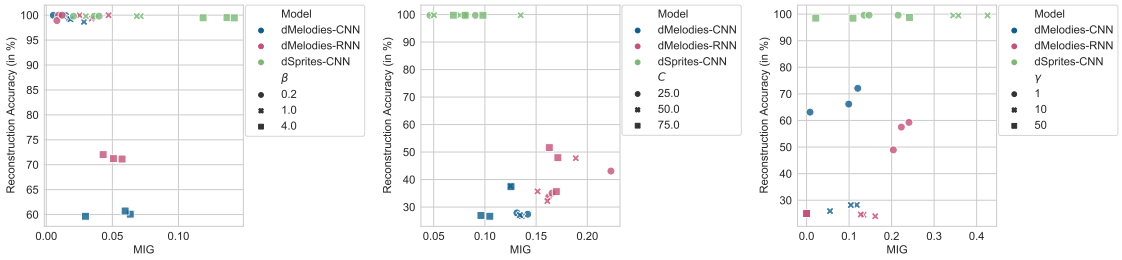
formance. For both the CNN and the hierarchical RNN-based VAE, the performance of all the different methods on dMelodies is comparable.

(c) Finally, the differences in the performance between the two datasets is compared. In terms of MIG and SAP, the performance for dSprites is slightly better (especially for Factor-VAE), while for Modularity, performance across both datasets is comparable. However, once again, the differences are not significant.

Looking at the disentanglement metrics alone, one might be tempted to conclude that the different methods are domain invariant. However, as the next experiments will show, there are significant differences.

6.3.3 Reconstruction Fidelity

From a generative modeling standpoint, it is important that, along with better disentanglement performance, we also retain good reconstruction fidelity. This is measured using the reconstruction accuracy shown in Figure 6.4. It is clear that all three methods fail to achieve a consistently good reconstruction accuracy on dMelodies. β-VAE gets an accuracy $\geq 90\%$ for some hyperparameter values (more on this in Section 6.3.4). However, both Annealed-VAE and Factor-VAE struggle to cross a median-accuracy of 40% (which would



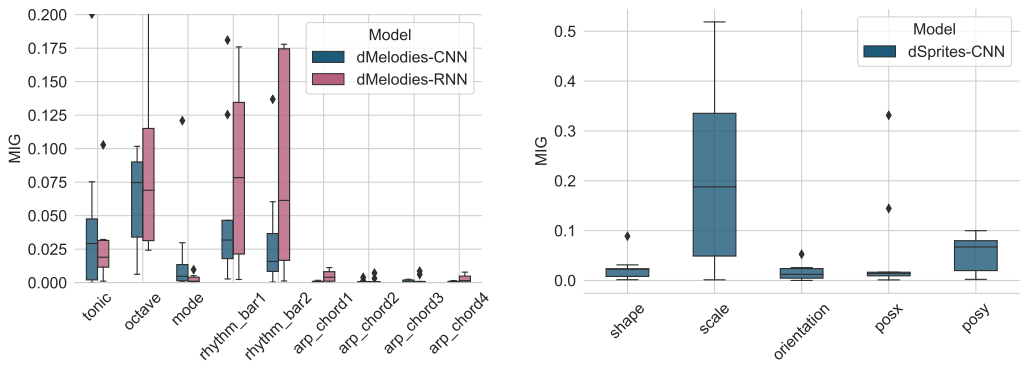
(a) β-VAE: Varying β (b) Annealed-VAE: Varying C (c) Factor-VAE: Varying γ

Figure 6.5: Effect of the hyperparameters on the different unsupervised disentanglement methods. Overall, improving disentanglement on dMelodies results in a severe drop in reconstruction accuracy. The dSprites dataset does not suffer from this drawback.

be unusable from a generative modeling perspective). The performance of the hierarchical RNN-based VAE is slightly better than the CNN-based architecture. In comparison, for dSprites, all three methods are able to consistently achieve better reconstruction accuracies.

6.3.4 Sensitivity to Hyperparameters

The previous experiments presented aggregated results over the different hyperparameter values for each method. Next, the individual impact of those hyperparameters is looked at. Figure 6.5 shows this in the form of scatter plots. The ideal models should lie on the top right corner of the plots (with high values of both reconstruction accuracy and MIG). Models trained on dMelodies are very sensitive to hyperparameter adjustments. This is especially true for reconstruction accuracy. For instance, increasing β for the β-VAE model improves MIG but severely reduces reconstruction performance. For Annealed-VAE and Factor-VAE there is a wider spread in the scatter plots. For Annealed-VAE, having a high capacity C seems to marginally improve reconstruction (especially for the recurrent VAE). For FactorVAE, increasing γ leads to a drop in both disentanglement and reconstruction. Contrast this with the scatter plots for dSprites. For all three methods, the hyperparameters seem to only significantly affect disentanglement performance. For instance, increasing β and γ (for β-VAE and FactorVAE, respectively) result in a clear improvement in MIG. More importantly, however, there is no adverse impact on reconstruction accuracy.



(a) *dMelodies* (b) *dSprites*

Figure 6.6: Factor-wise MIG for the β-VAE method on dMelodies (left) and dSprites (right).

6.3.5 Factor-wise Disentanglement

It is also important to evaluate how the individual factors of variation are disentangled. For this, the β-VAE model is considered since it has the highest reconstruction accuracy. Figure 6.6(a) shows the factor-wise *MIG* for both the CNN and RNN-based models. Factors corresponding to octave and rhythm are disentangled better. This is consistent with some recent research on disentangling rhythm [112, 163]. In contrast, the factors corresponding to the arpeggiation direction perform the worst. This might be due to their binary type. A similar analysis for the dSprites dataset in Figure 6.6(b) reveals better disentanglement for the scale and position based factors. Additional results are provided in Appendix C.2.

6.3.6 Discussion

As mentioned in Section 6.1, disentanglement techniques have been shown to be sensitive to the choice of hyperparameters and random seeds [89]. The results obtained in the experiments in this section using dMelodies seem to ascertain this even further. Methods which work well for image-based datasets do not extend directly to the music domain. When moving between domains, not only does one have to tune hyperparameters separately, but the model behavior may vary considerably when hyperparameters are changed. For instance,

reconstruction fidelity is hardly affected by hyperparameter choice in the case of dSprites while for dMelodies it varies significantly.

While sensitivity to hyperparameters is expected in neural networks, this is also one of the main reasons for evaluating methods on more than one dataset, preferably from multiple domains. Some aspects of the dataset design, especially the nature of the factors of variation, might have affected the experimental results. While the factors of variation in dSprites are continuous (except the shape attribute), those for dMelodies span different data-types (categorical, ordinal, and binary). This might make other types of models (such as VQ-VAEs [164]) more suitable. Another consideration is that some factors of variation (such as the arpeggiation direction and rhythm) affect only a part of the data. However, the effect of this on the disentanglement performance needs further investigation since the disentanglement performance for rhythm is much better than that for arpeggiation direction.

Another possible reason why unsupervised methods work to reasonable degree in the case of images but fail for music could be related to the differences in the perception of images and music. Music is a temporal signal, and small "errors" can lead to significantly lower perceptual ratings (also seen in the listening experiments in Section 3.5.4). In contrast, images are perceived more holistically and smaller discrepancies are often ignored.

6.4 Experiments using Supervised Methods

Results obtained in the previous section showed that unsupervised disentanglement methods do not work well on the dMelodies dataset. It is still unclear if it is possible to develop general domain-invariant disentanglement methods. Recent studies have claimed that some degree of supervision might be essential to learning disentangled representations [89, 102]. This section looks at how different supervised disentanglement methods perform on the dMelodies dataset.

6.4.1 Experimental Set-Up

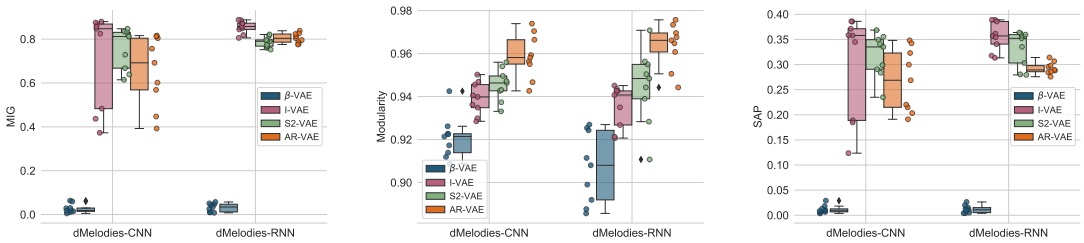
Three different supervised disentanglement methods are considered. The first method, referred to as I-VAE, is based on the regularization used by Adel et al. to [105]. This uses a separate single linear classifier attached to each regularized dimension of the latent space to predict the attribute classes. This is a suitable choice considering the categorical attributes in the dMelodies dataset. The second method is the S2-VAE [102] which was also used for comparison in Chapter 4. Finally, the AR-VAE framework is used as the third method. Since both AR-VAE and S2-VAE are designed for continuous attributes, the factors of variation are treated as continuous values by considering the class index of the category as the attribute value and then normalizing them to range in $[0, 1]$. For instance, the *Scale* attribute has 3 distinct options and hence, the possible values are $[0, \frac{1}{2}, 1]$ corresponding to the major, harmonic minor, and blues scales, respectively. Since, all the above methods use a regularizing hyperparameter γ corresponding to the regularization strength, three different values of $\gamma \in \{0.1, 1.0, 10.0\}$ are used.

The data representation, model architectures, and disentanglement metrics are kept the same as that in the previous section for unsupervised disentanglement. Again, for each of the above method (I-VAE, S2-VAE, AR-VAE), model (dMelodies-CNN, dMelodies-RNN), and hyperparameter ($\gamma \in \{0.1, 1.0, 10.0\}$) combination, 3 models with different random seeds were trained. Other training specifications were kept the same as in the previous section on unsupervised disentanglement.

6.4.2 Disentanglement

The disentanglement performance of the three supervised methods is compared against the β-VAE model in Figure 6.7. The following observations can be made:

(a) All three supervised methods clearly outperform the β-VAE across the three disentanglement metrics. The improvement is much higher for the *MIG* and *SAP* score which



(a) *MIG* (b) *Modularity* (c) *SAP Score*

Figure 6.7: Overall disentanglement performance (higher is better) of different supervised methods on dMelodies. Individual points denote results for different hyperparameter and random seed combinations. Results for β-VAE are also shown for comparison.

both measure the degree to which each factor of variation is encoded in only a single dimension of the latent space.

(b) Among the supervised methods themselves, the only clear difference is for *Modularity*, where AR-VAE outperforms the other two. For the other two metrics, all three methods perform comparably.

(c) Among the two different model architectures, the overall variance (dependence on random seeds and hyperparameters) is lower for the RNN-based model in *MIG* and *SAP* score. This might be attributed to the more complex architecture of dMelodies-RNN. Contrast this to Figure 6.3 where both model architectures showed comparable results with the different unsupervised methods.

Using supervision, therefore, leads to better overall disentanglement.

6.4.3 Reconstruction Fidelity

The reconstruction performance for the supervised methods is shown in Figure 6.8. Compared to β-VAE, all three supervised methods are able to get a high reconstruction accuracy which makes them usable from the generative modeling standpoint. Once again, like the disentanglement results, there are no considerable differences between the different meth-

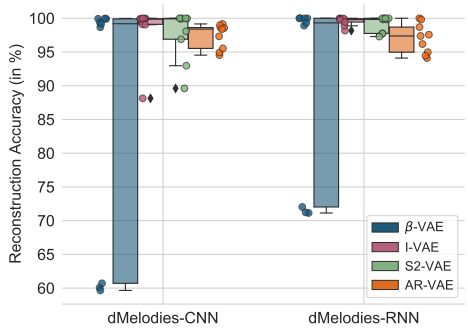
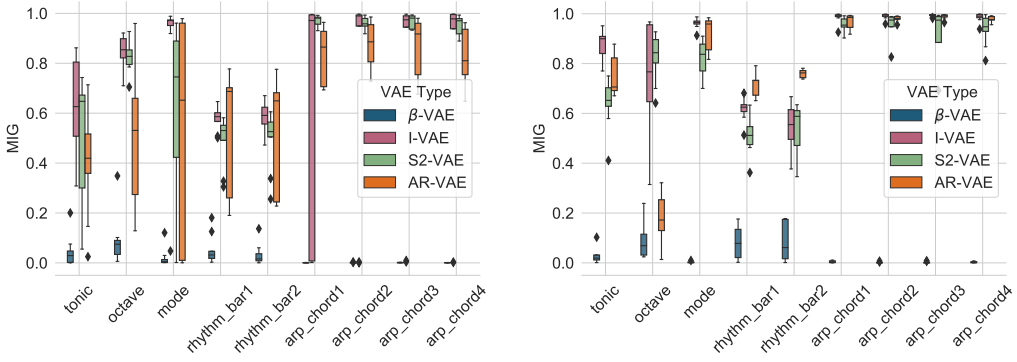


Figure 6.8: Overall reconstruction accuracies (higher is better) of the supervised methods on dMelodies. Individual points denote results for different hyperparameter and random seed combinations. Results for β-VAE are also shown for comparison.



(a) *dMelodies-CNN* (b) *dMelodies-RNN*

Figure 6.9: Factor-wise MIG for the different supervised methods on dMelodies. Results for β-VAE are also shown for comparison.

ods. For reconstruction accuracy, there also does not seem to be any noticeable difference between the different model architectures.

Overall, the improved disentanglement performance for the supervised methods is achieved without sacrificing the reconstruction quality.

6.4.4 Factor-wise Disentanglement

Factor-wise disentanglement is considered next and is shown in Figure 6.9. Compared to β-VAE, supervised methods are better at disentangling all the different factors of variation. Even the binary factors of variation seem to be disentangled properly. The overall variance

(dependence on hyperparameters and random seeds) across different factors is lower for the RNN-based models which was also seen in the corresponding experiment using unsupervised methods (see Section 6.3.5). The performance of the AR-VAE method, in particular, improves significantly with the RNN-based model. Compared to other models, S2-VAE seems to be more consistent and robust against hyperparameter and random seeds.

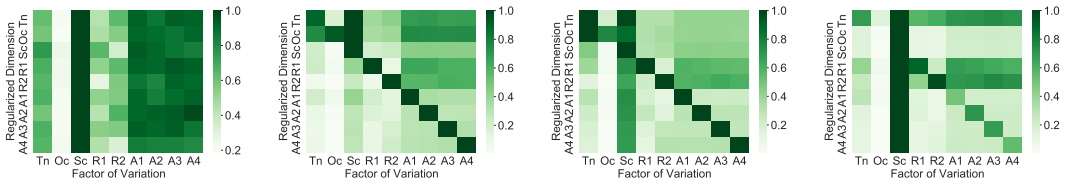
Overall, supervised methods show considerably improved performance at disentangling different types of attributes.

6.4.5 Attribute Disentanglement during Generation

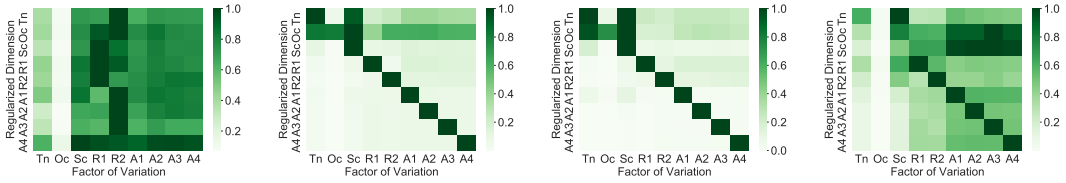
Considering that the supervised methods are able to obtain better disentanglement along with good reconstruction accuracy, the next step is to look at how effective these methods are for controlling different attributes. To measure this quantitatively, a process similar to that in Section 4.3.3 is followed, albeit with a few changes to take into account the discrete and categorical nature of the attributes. Given a data-point with latent vector $\mathbf{z}$, 6 different variations are generated by uniformly interpolating along the dimension r_l, where r_l is the regularized dimension for attribute a_l. The limits of interpolation are chosen based on the maximum and minimum latent code values obtained during encoding the validation data. For the β-VAE model, the dimension with the highest mutual information with the attribute is considered as the regularized dimension. An attribute change matrix $A \in \mathbb{R}^{L \times L}$, where L is the number of attributes, is computed using the following formulation:

$$A(m, n) = \sum_{i=1}^{6} [0 \neq |a_n(\mathbf{z}_i^m) - a_n(\mathbf{z})|] \tag{6.1}$$

where, $A(m, n)$ computes the net change in the n^{th} attribute as one traverses the dimension r_m (which is the regularized dimension for the m^{th} attribute), $[\cdot]$ represents the Iverson bracket (inverse Kronecker delta function), $a_n(\cdot)$ is the value of the n^{th} attribute, and $\mathbf{z}_i^m$ is the i^{th} interpolation of $\mathbf{z}$ obtained by traversing along the r_m dimension. This attribute change matrix is computed for each model type by averaging over a total of 1024 data-points



(a) β-VAE, CNN (b) I-VAE, CNN (c) S2-VAE, CNN (d) AR-VAE, CNN



(e) β-VAE, RNN (f) I-VAE, RNN (g) S2-VAE, RNN (h) AR-VAE, RNN

Figure 6.10: Net change in attribute values during traversal along the regularized dimension. The columns show the normalized net change in a particular attribute value as one traverses the regularized dimension for the attribute corresponding to the rows.

in the test-set and across all 3 random seeds (regularization hyperparameters are fixed at $\beta = 0.2$, $\gamma = 1.0$). The matrix is also normalized so that the maximum value across each row corresponds to one.

The results obtained for different methods and model architectures are shown in Figure 6.10. Ideal disentanglement should result in a plot with high values (dark green color) along the diagonal (there should be a change in the regularized attribute only) and lighter colors on the off-diagonal entries. The following observations can be made:

(a) Among the supervised methods, I-VAE and S2-VAE seem to perform better than AR-VAE. This can be seen in the lighter shades on the off-diagonal elements in the plots for I-VAE and S2-VAE. While the better performance of S2-VAE is expected since it is designed for categorical attributes, the poorer performance of AR-VAE in comparison to S2-VAE needs further investigation.

(b) Among the two different model architectures, RNN-based models perform better at task. This is in line with results from the previous experiments in this section (see Section 6.4.2 and 6.4.5).

(c) The β-VAE method clearly performs the worst where traversals along most dimensions change multiple attributes simultaneously.

(d) The *scale* attribute (3rd column) changes the most while traversing the regularized dimensions for the supervised methods. This indicates that all supervised methods struggle in generating notes conforming to particular scales.

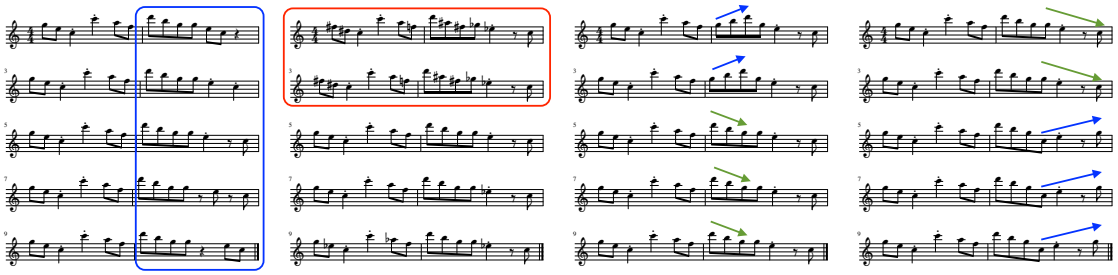
It is worth noting that while there was no considerable difference between the disentanglement performance (see Section 6.4.2) of the three methods on the test-data, S2-VAE and I-VAE show much better performance compared to AR-VAE in this experiment. Overall, these results show that supervised methods are also better at disentangling attributes during data generation which potentially makes them suitable for manipulating specific attributes. The next experiment presents a qualitative examination of this aspect.

6.4.6 Inspecting Latent Interpolations

This experiment deals with the qualitative examination of the data generated by the different supervised methods while traversing the latent space along the regularized dimensions. The expectation is that traversing along a regularized dimension should only cause changes in the corresponding attribute while leaving the other attributes unchanged. In addition, another desirable property is that the regularized attribute should change in a predictable manner.

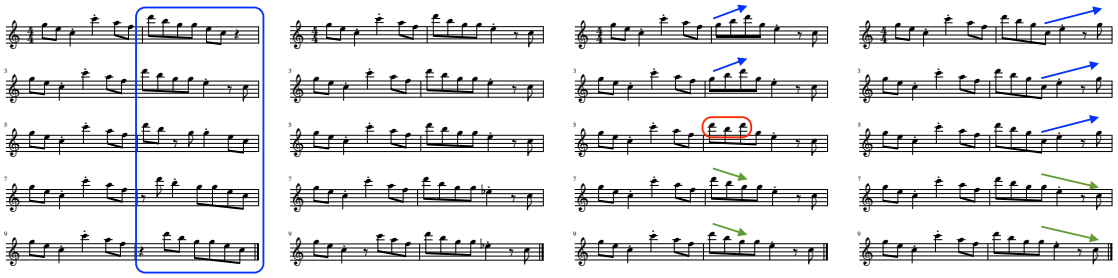
Figure 6.11 shows the results of the I-VAE method. For each sub-figure, each row corresponds to a melody generated by traversing along the regularized dimension for the attribute listed in the sub-figure caption. Similar results for S2-VAE and AR-VAE are shown in Figure 6.12 and Figure 6.13, respectively. The RNN-based model is chosen for all three methods compared in this experiment.

Across methods, most of the time, the melodies generated by traversing along regularized dimensions show changes in the corresponding attribute only. For instance, in Figures 6.11(a), 6.12(a), and 6.13(a), only the rhythm of the second bar changes while the rest of the melody stays intact. In Figure 6.11(c,d), the arpeggiation directions of the third



(a) Rhythm Bar 2 (b) Scale (c) Arp Chord 3 (d) Arp Chord 4

Figure 6.11: Generated data by traversing along regularized dimensions of a dMelodies-RNN model trained using the I-VAE method. For each sub-figure, each row corresponds to a data-point generated by traversing along the regularized dimension for the attribute listed in the sub-figure caption.

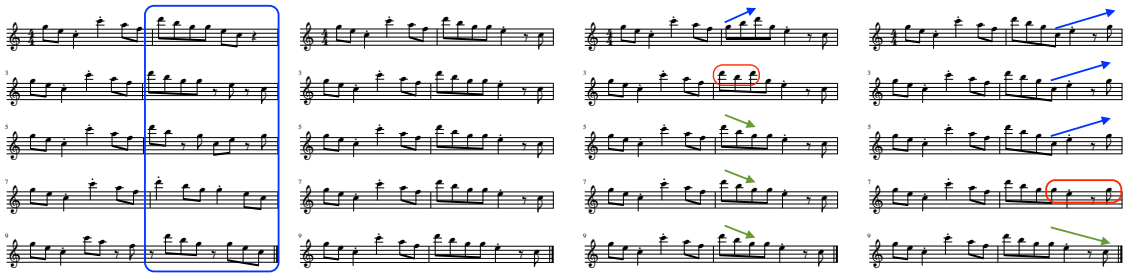


(a) Rhythm Bar 2 (b) Scale (c) Arp Chord 3 (d) Arp Chord 4

Figure 6.12: Generated data by traversing along regularized dimensions of a dMelodies-RNN model trained using the S2-VAE method. For each sub-figure, each row corresponds to a data-point generated by traversing along the regularized dimension for the attribute listed in the sub-figure caption.

and fourth chords are flipped, respectively. Also, in Figure 6.11(b), all the other attributes remain constant (rhythm, arpeggiation directions) while the pitches of the generated notes change to reflect different scales. While this is desirable, there a few important things to note.

First, the *scale* attribute seems hard to control. In Figure 6.13(b), for AR-VAE, the *scale* does not change at all. On the other hand, in Figure 6.11(b), for I-VAE, some of the generated melodies (the first two rows) do not conform to any of the scales present in the dataset. This difficulty in controlling the *scale* attribute was also seen in Section 6.4.5.



(a) Rhythm Bar 2 (b) Scale (c) Arp Chord 3 (d) Arp Chord 4

Figure 6.13: Generated data by traversing along regularized dimensions of a dMelodies-RNN model trained using the AR-VAE method. For each sub-figure, each row corresponds to a data-point generated by traversing along the regularized dimension for the attribute listed in the sub-figure caption.

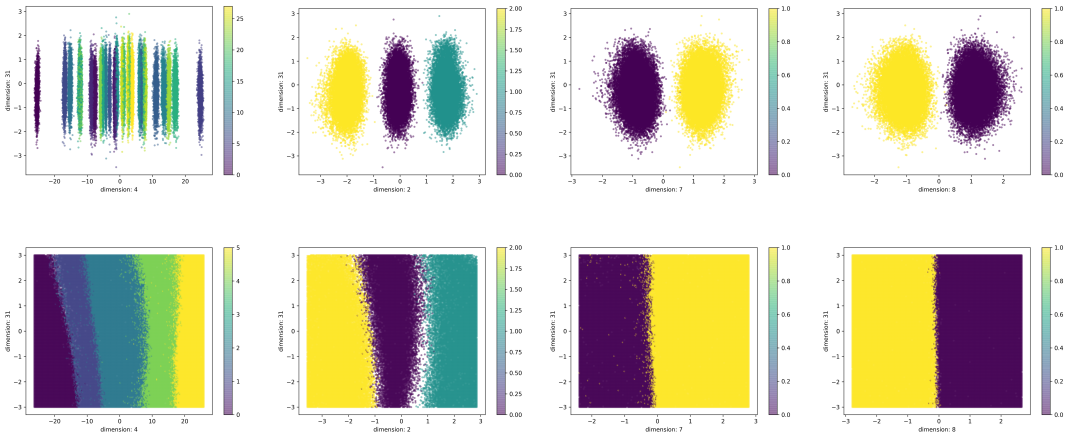
Second, traversals along regularized dimensions sometimes create melodies with attributes which are unseen in the training data (out-of-distribution). This is also seen for attributes other than *scale*. For instance, in the third row of Figure 6.12(c), the third chord has an unseen arpeggiation direction. It first goes down and then up. The same is observed in the second row of Figure 6.13(c).

Finally, for I-VAE, the direction of change for arpeggiation factors (see Figure 6.11(c,d)) is unpredictable. While the arpeggiation direction (of the third chord) goes from up to down in Figure 6.11(c), the direction (for the fourth chord) is flipped from down to up in Figure 6.11(d). This is due to the nature of the regularization used by I-VAE which is agnostic to the order of the categorical attributes. Contrast this to S2-VAE and AR-VAE, where the manner in which the regularized attribute changes can be pre-defined. For instance, the direction of arpeggiation will always go from up to down for these models (see Figures 6.12(c,d), 6.13(c,d)).

Additional plots showing the changes in all the attributes are shown in Appendix C.3.

6.4.7 Latent Space Visualization

This experiment visualizes the structure of the latent space with respect to the different attributes using data distribution and latent surface plots. The overall process to create these



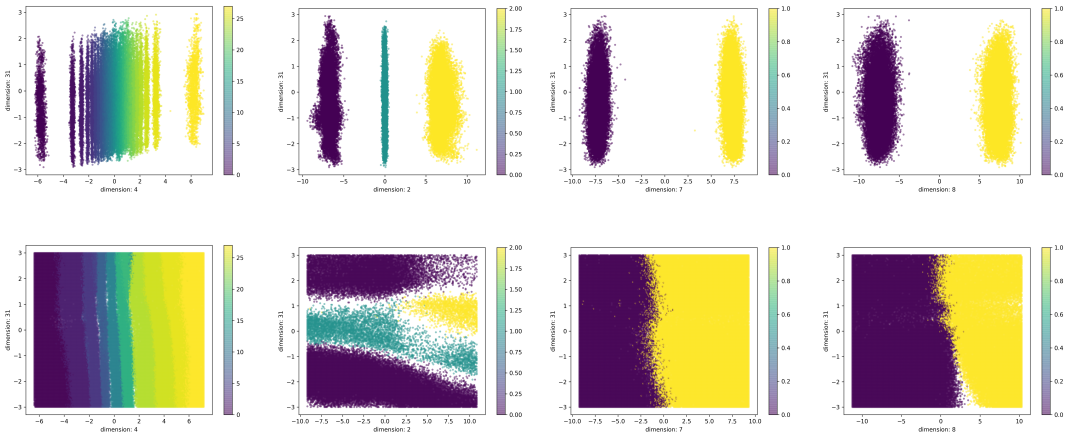
(a) *Rhythm Bar 2* (b) *Scale* (c) *Arp Chord 3* (d) *Arp Chord 4*

Figure 6.14: Encoder distribution (top row) and surface plots (bottom row) for the I-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.

visualizations is similar to Section 4.3.5. Figures 6.14, 6.15, and 6.16 show the results for I-VAE, S2-VAE, and AR-VAE, respectively. In each figure, the top row corresponds to data distribution plots (created by processing data-points from held-out test-set through the VAE encoder), and the bottom row shows the latent surface plots (created by generating data-points using the VAE decoder). The x-axis corresponds to the regularized dimension for an attribute and the y-axis corresponds to a non-regularized dimension. Attribute values are shown using different colors. The latent surface plots (bottom rows) in these visualizations differ from Section 4.3.5 in two ways:

(a) The range of traversal along the regularized dimension (x-axis) is determined based on the range obtained in the data distribution plots (top rows).

(b) The generated data-points sometimes have attribute values which are either not present in the training set or cannot be determined (e.g., the generated melody might not conform to any of the 3 possible scales used, or the arpeggiation direction might be neither up nor down). These 'undefined' or out-of-distribution attribute values are shown as empty spaces in the latent surface plots.



(a) *Rhythm Bar 2* (b) *Scale* (c) *Arp Chord 3* (d) *Arp Chord 4*

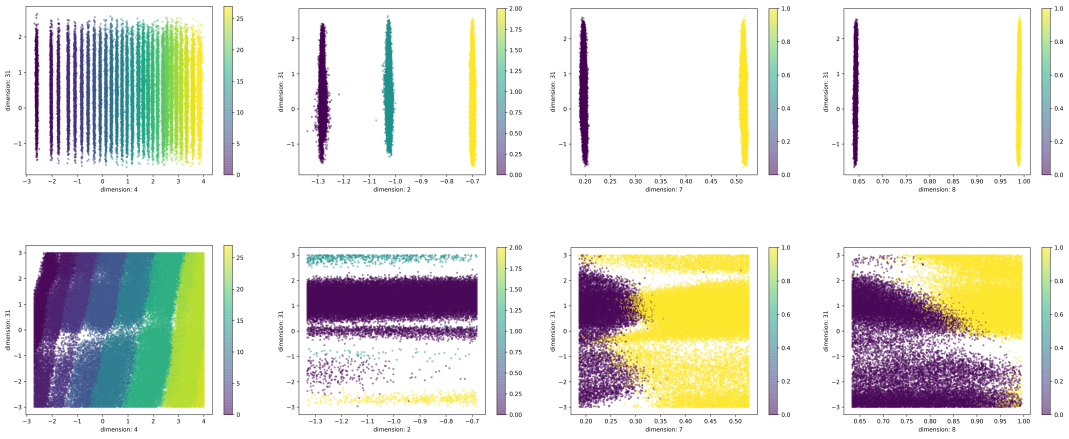
Figure 6.15: Encoder distribution (top row) and surface plots (bottom row) for the S2-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.

The following observations can be made:

(a) For all three methods, the data distribution plots (top rows) show a nice structure with respect to the attributes. This can be seen from the clear separation of colors (denoting individual attribute values) along the regularized dimensions. This explains the high disentanglement performance seen in Section 6.4.2. However, the methods differ considerably when the latent surface plots (bottom rows) are compared.

(b) For I-VAE (see Figure 6.14 bottom row), moving along the regularized dimension changes the corresponding attribute, while traversals along the non-regularized dimension have little effect. Although, as seen in Section 6.4.6, the nature of the change is not always predictable. For instance, while in Figure 6.14(c), the arpeggiation direction goes from up to down for a left to right traversal along the regularized dimension, this is reversed in Figure 6.14(d).

(c) S2-VAE (see Figure 6.14) has comparable performance as I-VAE across most attributes. In particular, the gradual change of color in Figure 6.14(a)(bottom) shows a high degree of controllability for the rhythm attributes. However, S2-VAE struggles to



(a) *Rhythm Bar 2* (b) *Scale* (c) *Arp Chord 3* (d) *Arp Chord 4*

Figure 6.16: Encoder distribution (top row) and surface plots (bottom row) for the AR-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.

disentangle the *scale* attribute. Traversing along the y-axis in Figure 6.14(c)(bottom) seems to have a considerable effect on the *scale* which is undesirable.

(d) The latent space of AR-VAE (see Figure 6.16) has the worst overall structure. Not only is the latent space not centered around the origin (see the top row of Figure 6.14(b,c,d)) for many attributes, but the degree of controllability is also poor. For instance, the *scale* attribute does not change at all along the regularized dimension (see Figure 6.14(b)(bottom)). In addition, the large amount of empty space in the surface plots shows that many of the generated data-points have an out-of-distribution attribute value.

(e) The empty regions in the latent spaces (as seen across all methods, albeit in different amounts) show that while these methods are able to train the encoders to disentangle different attributes, they end up creating *holes* or vacant regions in the latent space where the behavior of the decoder is unpredictable. This hinders the controllable manipulation of data attributes during generation.

Additional plots for the other attributes are shown in Appendix C.3.

6.4.8 Discussion

The results of the experiments in this section show that, on average, supervised methods for disentanglement perform better than unsupervised methods. This is expected since the former use attribute-specific information during training to guide the model towards learning better representations. Unsupervised methods, on the other hand, do not have access to this information and hence, do not perform as well.

Among the supervised methods, there are no major differences in terms of the disentanglement metrics in Section 6.4.2. However, disentanglement during data generation (discussed in Sections 6.4.5, 6.4.6, and 6.4.7) differs significantly between the methods. This is reflected in smoother interpolations for the I-VAE method where traversals along regularized dimensions lead to changes in the corresponding attributes. In contrast, for the other methods (particularly for AR-VAE), traversals often created data-points with unpredictable attributes. These differences suggest that while any of the above methods can be used for learning effective representations for discriminative tasks, not all methods are suitable from the perspective of controllable generation. Objectively evaluating controllability requires a different experimental design. In addition, it also necessitates the ability to compute the value of the different attributes from the generated data. Note that it is not quite straightforward to conduct an experiment similar to that used in Sections 6.4.5, 6.4.6, and 6.4.7 for the dSprites dataset because it is difficult to compute the attribute values (such as position, scale, shape) of newly generated data.

The experiments in this section also address the questions raised in Chapter 4 regarding the performance of AR-VAE on categorical attributes. While AR-VAE still performs much better than β-VAE even on categorical attributes, the design of the regularization loss makes it inferior to other supervised learning methods such as I-VAE. Once again, this is not reflected in any of the disentanglement metrics but becomes clear in experiments which evaluate the degree of controllability by inspecting data generated using latent traversals and interpolations.

6.5 Conclusion

This chapter presented a systematic investigation of disentanglement learning in the context of symbolic music by introducing a new dataset tailored to the task. The proposed *dMelodies* dataset comprises more than 1 million data points of 2-bar melodies. The dataset is constructed based on fixed rules that maintain independence between different factors of variation, thus enabling researchers to use it for studying disentanglement learning.

Several experiments are conducted on the dMelodies dataset using popular unsupervised and supervised disentanglement learning methods. The first set of experiments using unsupervised learning methods show that such methods do not achieve performance comparable to those obtained on the analogous image-based dSprites dataset. In addition, unsupervised methods struggled to maintain a high degree of disentanglement along with good reconstruction performance. These methods are also not able to deal efficiently with binary and other categorical factors of variation present in the dMelodies dataset. These results showcase that further research is needed to develop domain-invariant unsupervised algorithms for disentanglement learning.

Supervised learning methods perform much better at the task. Several experiments using supervised methods show that not only are these methods able to perform better at learning disentangled representations, but they are also far superior in terms of maintaining a high reconstruction accuracy and better controlling the attribute values during latent traversals and interpolations. These results are promising and reinforce the idea that supervised learning methods might be the way forward for improving the controllability of music generation models.

It is worth noting that all the experiments using supervised methods in this chapter used the labels from the entire training set. An interesting direction for future studies could be to extend these experiments to the semi-supervised paradigm by using a limited number of labels obtained from only a fraction of the training set [102]. This would further increase the

confidence in applying these methods to real-world data where obtaining label information for the entire dataset might be either too costly or simply impossible.

While dMelodies is a great starting point for experimenting with disentanglement learning for music, one criticism could be that it does not reflect real-world music. The melodies used are relatively simplistic and only consider arpeggiations over a very basic chord pattern. Most of the real-world melodies are not composed in this manner. The factors of variations are also contrived in a manner that forces independence among them. Also, some of the factors may not seem extremely relevant from a controllable music generation perspective. For instance, manipulating the octave or tonic of a melody is a rather trivial task and is not something that one might want a deep generative model to learn explicitly. In addition, the dataset ignores a vast majority of musical content by restricting itself to monophonic melodies. In spite of these limitations, dMelodies addresses a critical whitespace in the field of music representation learning and seeks to serve the same purpose as dSprites (and other artificially generated image-based datasets). The dSprites dataset is also created using contrived factors of variation. The resulting images are highly simplistic and do not reflect real-world images. However, the simple nature of the dataset along with the independent factors of variation has allowed dSprites to be used as a benchmarking dataset to compare and evaluate different disentanglement learning methods. Similar to dSprites, the experiments using dMelodies presented in this chapter provide useful insights into how disentanglement learning algorithms work on music data. This not only motivates the need for further research on developing better disentanglement learning methods but also coming up with new datasets that are better designed and reflect real-world music in a meaningful way.

CHAPTER 7
CONCLUSION

Developments in the field of automatic music creation systems have increasingly relied on neural network-based deep generative models. The ability of such models to learn complex patterns from large collections of data has enabled their widespread use in a variety of automatic music composition tasks. However, in spite of the commendable success of such systems, they have some serious shortcomings. The music composition process of such systems is often unidirectional and does not provide users with any meaningful opportunities for interaction. In addition, most systems work as a black-box and do not extend any control over different attributes of the generated music. This places severe limitations on the practical usability of such systems for music creation. In this thesis, I present different methods which seek to address these challenges of *interactivity* and *controllability* faced by deep generative models of music.

The primary motivation behind this thesis has been to leverage low-dimensional latent representations which are learnt by compressing information from high-dimensional data. The methods proposed in this thesis rely on supervised learning methods to manipulate such latent representations in order to: *(a)* effectively utilize the information already available in such latent spaces, and/or *(b)* force the latent representation to provide explicit control over particular attributes of the generated music.

7.1 Summary

In Section 1.5, I formulated the primary research questions that this thesis attempts to address. This section summarizes the experimental findings across Chapters 3-6 and contextualizes them against those research questions.

(RQ1) **How can information in latent representations of deep generative models be better used for controlling music generation?**

Chapter 3 addressed RQ1 by presenting a method to leverage the information present in the latent space of a trained VAE model. This was applied to the task of music inpainting, i.e, connecting two excerpts of music. Given a VAE trained to generate single measures of music, the proposed method used a set of RNNs to learn complex trajectories in the latent space of the VAE to accomplish the music inpainting task. The results show that latent representations can be used to implicitly model high-level musical information such as rhythm, structure, and repetitions.

Chapter 5 also addressed RQ1 but focused on providing explicit control over low-level musical attributes by leveraging the information already present in the latent spaces of vanilla-VAEs. In order to achieve this, the proposed LAR-VAE method combined the Lens framework [105] with the regularization method proposed in Chapter 4. The results show that while this method works on simpler image-based datasets, it does not work on the musical data to the same extent. There is either some information loss or a high degree of entanglement in vanilla-VAE latent spaces which prevents learning explicit and independent control over the musical attributes.

(RQ2) **To what extent can specific musical attributes be encoded in the latent spaces of music generation models?**

Chapter 4 addressed RQ2 by introducing a novel regularization method to explicitly encode selected low-level continuous-valued musical attributes such as *note density* and *rhythmic complexity* along individual dimensions of the latent space. The resulting AR-VAE model is able to learn structured latent spaces where individual attributes of the data can be explicitly and independently controlled by simply traversing along the dimensions of the latent space along which they are regularized.

Chapter 6 answered RQ2 with respect to categorical attributes. The experiments using

different supervised regularization methods show that while the AR-VAE regularization method from Chapter 4 does not perform as well, there are other learning methods that are more suitable for encoding and controlling categorical attributes in a meaningful manner.

(RQ3) **To what degree can individual musical attributes be disentangled in the latent space?**

Chapters 4 to 6 all addressed RQ3 in different ways. For measuring the degree of disentanglement, various objective metrics were used. This was done from the perspective of both unseen test-data as well as newly generated data. In addition, other qualitative measures (such as inspecting the generated data and visualizing the latent spaces) were also employed.

Chapter 4 compared disentanglement with respect to continuous musical attributes across different supervised and unsupervised methods. The results show that AR-VAE outperforms all the other methods and is able to learn latent representations which are superior at disentangling continuous musical attributes.

Chapter 5 extended the experiments in Chapter 4 by measuring the degree of disentanglement of different continuous attributes in the transformed latent space learnt using the proposed non-linear LAR-VAE transformation. This was compared with AR-VAE and the unsupervised β-VAE method. The results show that while disentanglement performance improves in the transformed latent space, it is still inferior to AR-VAE. Certain attributes (e.g., *note density*) are easy to disentangle, whereas others (e.g., *pitch range*) are much harder to deal with. In addition, any direct and indirect correlation between individual attributes also plays a crucial role in preventing effective disentanglement.

Finally, Chapter 6 addressed RQ3 by presenting a systematic study on music disentanglement. This was done by first introducing an algorithmically generated sym-

bolic music dataset designed specifically for the task. Then, a systematic study was conducted to compare and contrast the performance of several unsupervised and supervised learning methods on this dataset. An in-depth analysis was presented on how different methods fare with respect to the different attributes both in terms of disentanglement and controllability during music generation. The results show that unsupervised methods for disentanglement learning do not work well on music data. On the other hand, supervised methods perform significantly better.

7.2 Contributions

The main contributions of this thesis are tools for enabling interactive and controllable music generation applications. The proposed methods range from implicit modeling of high-level musical attributes (such as musical structure and repetition) to providing explicit control over low-level attributes (such as melodic contour and note density). This thesis also thoroughly investigated the problem of disentanglement learning in the context of symbolic music.

(a) **InpaintNet for Interactive Score Inpainting**

The first major contribution of this thesis is the InpaintNet model which addresses the problem of sequential generation in deep generative models. The model is capable of duly taking into account both past and future musical contexts to perform inpainting in a non-sequential and interactive manner. Comparison with competitive baseline systems operating on raw symbolic data shows the superior performance of the InpaintNet model across different objective and subjective tests. The main contribution of the InpaintNet model is in showing that implicit musical information, encoded in the latent representations learnt by VAEs, can be leveraged by using neural network-based learning methods. This also inspires the need to learn better latent representations from which such information can be easily extracted for different music generation tasks [137]. The InpaintNet approach of working with latent vectors, as opposed to raw data, can also be potentially used to generate music with better long-term

structure.

(b) **AR-VAE for Explicit Control over Musical Attributes**

The second contribution is the AR-VAE framework which allows a VAE-based generative model to explicitly control low-level continuous-valued musical attributes by simple traversals in the latent space. The proposed regularization method has other key advantages that it: *(a)* has a simple formulation with few hyperparameters, and *(b)* is completely agnostic to how the attributes are computed/obtained. Not only does AR-VAE achieve superior performance in the disentanglement of musical attributes, but it also performs well at providing independent and linear control over the attributes. While the experiments in this thesis focus on monophonic music, a recent study by other researchers using polyphonic piano music also validates the state-of-the-art performance of AR-VAE against other regularization and conditioning-based methods [143].

(c) **LAR-VAE for Disentangling Entangled Latent Representations**

The third contribution is the LAR-VAE framework which tries to disentangle the information already present in latent spaces of vanilla-VAEs. This approach relies on learning a non-linear invertible transformation from an existing (entangled) base latent space to a secondary transformed latent space where different attributes could be disentangled. This approach has several benefits: *(a)* it can be implemented even if only part of the data has information about the attributes, *(b)* it avoids the need to retrain the base model while adding new attributes, *(c)* the reconstruction accuracy of the base model is not affected, and *(d)* multiple transformations can be learnt from the same latent space to work for different attributes. While the LAR-VAE framework performs better than the unsupervised β-VAE model, the performance is inferior to AR-VAE models trained using raw data. In spite of this inferior performance, the experiments provide useful insights which will help guide future research in music

disentanglement learning.

(d) **dMelodies for Systematizing Music Disentanglement Learning**

The final major contribution of this thesis is in the field of disentanglement learning in the context of symbolic music. First, it introduces the dMelodies dataset. This artificially generated dataset consists of simple 2-bar monophonic melodies. The different design choices (e.g., using orthogonal attributes) make this dataset suitable for conducting systematic objective analyses of disentanglement learning methods. In the absence of any other benchmarking dataset for music disentanglement, dMelodies helps fill important whitespace. Second, a systematic study using different unsupervised and supervised methods is conducted. The results of this study clearly show the benefits of using supervised methods for disentanglement learning. In addition, the experiments also bring out the difference between learning a disentangled representation (which can be useful for discriminative tasks) and learning a controllable generative model (which can manipulate attributes independently). Overall, the dMelodies dataset and the accompanying benchmarking study will serve as a useful starting point for future research.

(e) **Other Contributions**

While the proposed methods in this thesis are targeted towards systems for interactive and controllable music creation, they are designed with a general machine learning framework in mind. Consequently, they can be applied to other domains in the field of computational creativity as well. For instance, the approach of using RNNs to learn trajectories in the latent space, which was used for the InpaintNet model, can be applied to other tasks such as text generation. The potential of AR-VAE in manipulating continuous attributes for images is already demonstrated in the different experiments using the image-based datasets across Chapters 4 and 5.

7.3 Avenues for Future Research

There are different ways in which each of the individual methods presented in Chapters 3 through 6 can be improved even further. These avenues are already discussed at the end of the respective chapters. From the perspective of controlling musical attributes using latent representations, there are two major directions covered below which need to be explored further.

7.3.1 Improving Interpolations in Latent Spaces

Some of the experiments in Chapters 4 and 6 show that there is a clear disconnect between disentanglement in the latent space and controllability during generation. For instance, even though the AR-VAE models show very good performance in terms of different disentanglement metrics (see Section 4.3.1), the interpolation-based experiments (see Sections 4.3.3 and 4.3.4) show that trying to control one attribute using latent traversals can lead to unpredictable changes across different attribute values. Consequently, a disentangled representation is not a sufficient condition to guarantee effective control over the generation process.

As discussed in Section 6.4.7, one possible reason for this could be that there are holes (vacant regions) in the latent spaces from which the VAE-decoder is unable to generate realistic samples. Similar observations have also been made for other data domains which rely on discrete data such as text [165]. There are a few promising directions to address this problem. One option is to constrain the latent space to conform to a specific manifold and perform manipulations within this manifold [165, 153]. An alternative direction could be to learn specific transformation paths within the existing latent manifold so as to avoid these holes [166].

7.3.2 Controlling High-Level Musical Attributes

The AR-VAE regularization method proposed in Chapter 4 has been shown to work quite well for low-level attributes. However, high-level musical attributes (e.g., tension, arousal, style) present unique challenges. These attributes evolve over longer time-scales and are extremely subjective in nature [42]. Thus, it is difficult to obtain large amounts of data to use fully-supervised regularization-based methods to provide control over these attributes.

Some recent works have tried using unsupervised and semi-supervised methods to model these attributes. For instance, Choi et al. use a transformer-based auto-encoder to implicitly learn an embedding corresponding to performance style [167]. A more interesting approach was used by Tan and Herremans where a high-level attribute (arousal) was modeled by using a combination of different low-level attributes (note density and rhythm density) [143]. This approach presents a promising direction for future investigations for modeling high-level musical attributes. Further research is needed to figure out the exact nature of the relationship between different musical attributes.

In conclusion, the models presented in this thesis allow finer control over the musical output of deep generative models and allow greater opportunities for user interaction. Together, they are solid steps towards making deep generative models for music more useful in practical compositional scenarios. I hope that the methods presented will lead to significant advances in the field of computer-assisted music creation. From a broader perspective, the individual techniques and results obtained across the different experiments have the potential to positively affect broader domains such as representation and disentanglement learning, generative modeling, and computational creativity at large.

Appendices

APPENDIX A

LATENT SPACE REGULARIZATION (EXPERIMENTAL DETAILS)

This appendix provides additional details and results for different experiments carried out using the AR-VAE method proposed in Chapter 4.

A.1 Computations of Musical Attributes

The data representation scheme is the same as the one used in Section 3.3.3. Each monophonic measure of music M is a sequence of N symbols $\{m_t\}, t \in [0, N)$, where $N = 24$. The set of symbols consists of note names (e.g., A#, Eb, B, C), a continuation symbol '_', and a special token for Rest. The computation steps for the individual attributes are as follows:

(a) Rhythmic Complexity (r): This attribute measures the rhythmic complexity of a given measure. To compute this, a complexity coefficient array $\{f_t\}, t \in [0, N)$ is first constructed which assigns weights to different metrical locations based on Toussaint's metrical complexity measure [142]. Metrical locations which are on the beat are given low weights while locations which are off-beat are given higher weights. The attribute is computed by taking a weighted average of the note onset locations with the complexity coefficient array f. Mathematically,

$$r(M) = \frac{\sum_{t=0}^{N-1} \mathrm{ONSET}(m_t).f_t}{\sum_{t=0}^{N-1} f_t}, \tag{A.1}$$

where $\mathrm{ONSET}(\cdot)$ detects if there is a note onset at location t, i.e., it is 1 if m_t is a note name symbol and 0 otherwise.

(b) Pitch Range (p): This is computed as the normalized difference between the maximum

and minimum MIDI pitch values:

$$p(M) = \frac{1}{R}\left(\max_{t\in[0,N)}(\text{MIDI}(m_t)) - \min_{t\in[0,N)}(\text{MIDI}(m_t))\right), \tag{A.2}$$

where $\text{MIDI}(\cdot)$ computes the pitch value in MIDI for the note symbol. The MIDI pitch value for Rest and '_' symbols are set to zero. The normalization factor R is based on the range of the dataset.

(c) Note Density (d): This measures the count of the number of notes per measure normalized by the total length of the measure sequence:

$$d(M) = \frac{1}{N}\sum_{i=0}^{N-1}\text{ONSET}(m_t), \tag{A.3}$$

where $\text{ONSET}(\cdot)$ has the same meaning as in Eq. (A.1) above.

(d) Contour (c): This measures the degree to which the melody moves up or down and is measured by summing up the difference in pitch values of all the notes in the measure. Mathematically,

$$c(M) = \frac{1}{R}\sum_{t=0}^{N-2}\left[\text{MIDI}(m_{t+1}) - \text{MIDI}(m_t)\right], \tag{A.4}$$

where $\text{MIDI}(\cdot)$ and R have the same meaning as in Eq. (A.2) above.

A.2 Implementation Details

Image-based Models: For the image-based models, a stacked convolutional VAE architecture is used. The encoder consists of a stack of N 2-dimensional convolutional layers followed by a stack of linear layers. The decoder mirrors the encoder and consists of a stack of linear layers followed by a stack of N 2-dimensional transposed convolutional layers. The architecture details are given in Tables A.1 and A.2.

Table A.1: Table showing architecture details of VAE used for the dSprites dataset in the AR-VAE experiments. In the Encoder Linear Stack, the last layer has two parallel linear layers for computing the mean and log standard deviation of the latent vectors respectively. Conv: 2-dimensional convolutional layer, TrConv: 2-dimensional transposed convolutional layer, i: input channels, o: output channels, k: kernel size, s: stride, p: padding, d: dropout probability, SELU: Scaled Exponential Linear Unit [168], ReLU: Rectifier Linear Unit.

Model Type	dSprites VAE
Encoder Convolutional Stack	4-layer Convolutional Network: Conv(i=1, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU
Encoder Linear Stack	3-layer Linear Network: Linear(i=512, o=256) + ReLU Linear(i=256, o=256) + ReLU Linear(i=256, o=10) $\times$ 2 (in parallel)
Decoder Linear Stack	3-layer Linear Network: Linear(i=10, o=256) + ReLU Linear(i=256, o=256) + ReLU Linear(i=256, o=512) + ReLU
Decoder Convolutional Stack	4-layer Transposed Convolutional Network: TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=1, k=4, s=2, p=1)

Music-based Models: For the music-based models, the MeasureVAE architecture from Figure 3.4 is used with the architecture details shown in Table A.3.

Training Details: All models for the same dataset are trained for the same number of epochs (models for both image-based datasets and Bach Chorales were trained for 100 epochs, models for the Folk Music dataset were trained for 30 epochs). The optimization was carried out using the ADAM optimizer [132] with a fixed learning rate of $1\text{e}{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1\text{e}{-8}$.

Table A.2: Table showing architecture details of VAE used for the Morpho-MNIST dataset in the AR-VAE experiments. Other details are the same as Table A.1.

Model Type	Morho-MNIST VAE
Encoder Convolutional Stack	3-layer Convolutional Network: Conv(i=1, o=64, k=4, s=1, p=0) + SELU + Dropout(d=0.5) Conv(i=64, o=64, k=4, s=1, p=0) + SELU + Dropout(d=0.5) Conv(i=64, o=8, k=4, s=1, p=0) + SELU + Dropout(d=0.5)
Encoder Linear Stack	2-layer Linear Network: Linear(i=2888, o=256) + SELU Linear(i=256, o=16) $\times$ 2 (in parallel)
Decoder Linear Stack	2-layer Linear Network: Linear(i=16, o=256) + SELU Linear(i=256, o=2888) + SELU
Decoder Convolutional Stack	3-layer Transposed Convolutional Network: TrConv(i=8, o=64, k=4, s=1, p=0) + SELU + Dropout(d=0.5) TrConv(i=64, o=64, k=4, s=1, p=0) + SELU + Dropout(d=0.5) TrConv(i=64, o=1, k=4, s=1, p=0)

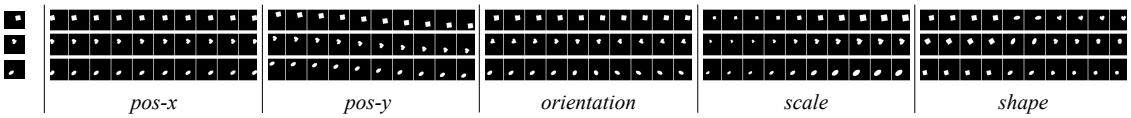


Figure A.1: Manipulating attributes for three different shapes from the dSprites sprites dataset using AR-VAE.

A.3 Additional Experimental Results

A.3.1 Latent Interpolations

Some additional latent interpolation examples (see Section 4.3.4) for the image datasets are shown in Figures A.1 and A.2. The interpolations for each attribute are generated by traversing along the corresponding regularized dimension for the original images shown on the extreme left. The figures show that attributes can be manipulated independently for both datasets. For Morpho-MNIST in particular, the digit identity is retained in most cases during interpolations.

Table A.3: Table showing configurations of the MeasureVAE architecture used for AR-VAE experiments. n: number of layers, i: input size, o: output size, h: hidden size, d: dropout probability, SELU: Scaled Exponential Linear Unit [168], ReLU: Rectifier Linear Unit, GRU: Gated Recurrent Units [169].

Measure VAE	
Embedding Layer	i=dict size, o=10
EncoderRNN	n=2, i=10, h=128, d=0.5 , type=GRU
Linear Stack 1 Linear Stack 2	i=512, o=32, n=2, non-linearity=SELU
BeatRNN	n=2, i=1, h=128, d=0.5, type=GRU
TickRNN	n=2, i=138, h=128, d=0.5, type=GRU
Linear Stack 3	i=128, o=256, n=1, non-linearity=SELU
Linear Stack 4	i=128, o=dict size, n=1, non-linearity=ReLU

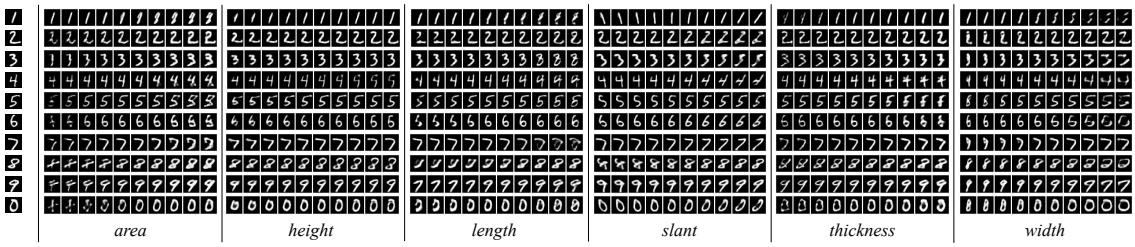
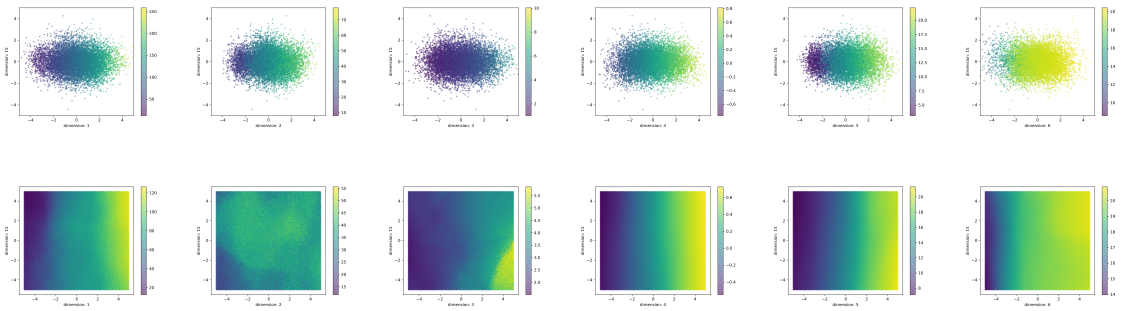


Figure A.2: Manipulating attributes for three different shapes from the Morpho-MNIST dataset using AR-VAE.

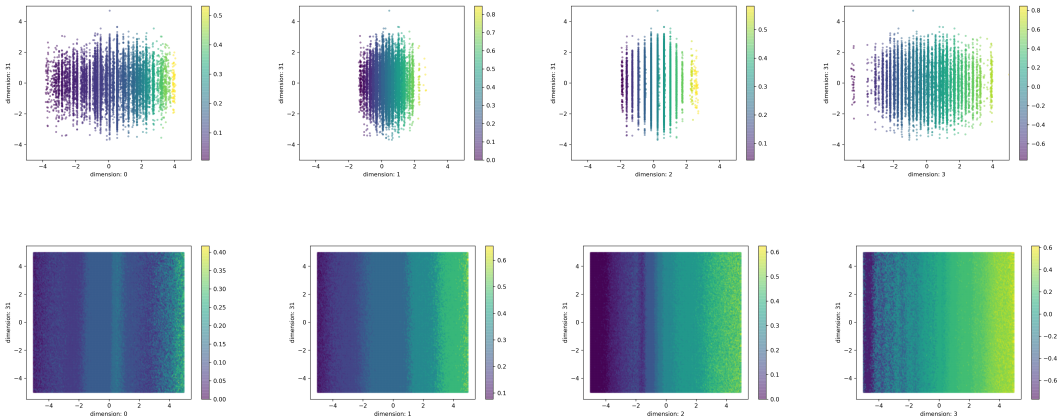
A.3.2 Latent Space Visualizations

Additional results for latent space visualizations (see Section 4.3.5) using encoder distribution plots and decoder latent surface plots are shown for all the attributes for the Morpho-MNIST and Folk Music datasets in Figure A.3 and A.4, respectively.



(a) *Area* (b) *Length* (c) *Thickness* (d) *Slant* (e) *Width* (f) *Height*

Figure A.3: Encoder distribution (top row) and latent surface plots (bottom row) for the Morpho-MNIST dataset. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Lighter colors indicate higher attribute values.



(a) *Rhythmic Complexity* (b) *Pitch Range* (c) *Note Density* (d) *Contour*

Figure A.4: Encoder distribution (top row) and latent surface plots (bottom row) for the Folk Music dataset. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Lighter colors indicate higher attribute values.

APPENDIX B

LATENT SPACE TRANSFORMATION (EXPERIMENTAL DETAILS)

B.1 Glow Model Architecture

The Glow-based [148] LAR-VAE model used for the experiments in Chapter 5 is composed of a sequence of K flow steps. Each flow step consists of 3 components: *(a)* an activation normalization layer, *(b)* an invertible 1×1 convolution layer (InvConv), and *(c)* an affine coupling layer (). Detailed descriptions of these layers can be found in the original paper by Kingma and Dhariwal [148]. A short summary of the forward function, inverse function, and log-determinants of the individual layers are reproduced here for easy reference in Table B.1.

Table B.1: Details of Glow model flow-step layers. Both $\mathbf{x}$ and $\mathbf{y}$ are tensors of shape $z \times 1$, i denotes the spatial index of the tensors, $\mathbf{W}$ is the weight matrix for the convolution, MLP is a non-linear multi-Layer perceptron with a single hidden layer of size H, $\mathrm{split}(\cdot)$ divides its input tensor into two equally-sizes tensors, and $\mathrm{concat}(\cdot)$ is a simple concatenation operation.

Description	**Forward**	**Inverse**	**Log-determinant**
Activation normalization	$\mathbf{y}_i = \mathbf{s} \odot \mathbf{x}_i + \mathbf{b}$	$\mathbf{x}_i = (\mathbf{y}_i - \mathbf{b})/\mathbf{s}$	$z \cdot \sum \log \mathbf{s}$
Invertible 1×1 conv	$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i$	$\mathbf{x}_i = \mathbf{W}^{-1}\mathbf{y}_i$	$z \cdot \log \lvert\det(\mathbf{W})\rvert$
Affine coupling layer	$\mathbf{x}_a, \mathbf{x}_b = \mathrm{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathrm{MLP}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathrm{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathrm{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathrm{MLP}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathrm{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\sum \log \lvert\mathbf{s}\rvert$

For the experiments in this chapter, the number of flow-steps (K) is fixed at 4. For the affine coupling layers, Multi-Layer Perceptrons (MLP) with a hidden size of 32 are used. Note that architectures using spline-based flows [170] were also experimented with initially but did not lead to good results.

APPENDIX C

DISENTANGLEMENT LEARNING (EXPERIMENTAL DETAILS)

This appendix provides additional details and results for different experiments on music disentanglement carried out using the dMelodies dataset proposed in Chapter 6.

C.1 Model Architectures

dMelodies-CNN: The architecture is similar to those used for several image-based VAEs, except that 1-D convolutions are used instead of 2-D convolutions. Details are given in table C.1.

dMelodies-RNN: This is based on the MeasureVAE architecture from Figure 3.4. Specifically, the MeasureVAE architecture from Table 3.1 is used with the following changes: (a) the latent size dimensionality is reduced to 32, (b) the hidden size of the encoder and decoder RNNs are reduced to 64, (c) the beatRNN is converted to a measure-level which is unrolled 2 times (there are 2-bars in each data point), and (d) the tickRNN is unrolled 8 times (there are 8 ticks per bar).

dSprites-CNN: A CNN-based architecture with the details provided in table C.2 is used.

C.2 Additional Results for Unsupervised Disentanglement

Additional factor-wise disentanglement plots for different methods and datasets are shown in Figure C.1.

Table C.1: dMelodies-CNN model architecture details. In the Encoder Linear Stack, the last layer has two parallel linear layers for computing the mean and log standard deviation of the latent vectors respectively. Conv: 1-D convolutional layer, TrConv: 1-D transposed convolutional layer, i: input channels, o: output channels, k: kernel size, s: stride, p: padding, SELU: Scaled Exponential Linear Units [129].

Model Type	dMelodies-CNN
Note Embedding Layer	Embedding(62, 10)
Encoder Convolutional Stack	4-layer Convolutional Network: Conv(i=10, o=16, k=4, s=2, p=1) SELU + Dropout(0.1) Conv(i=16, o=32, k=4, s=2, p=1) SELU + Dropout(0.1) Conv(i=32, o=64, k=4, s=2, p=1) SELU + Dropout(0.1) Conv(i=64, o=128, k=4, s=2, p=1) SELU + Dropout(0.1)
Encoder Linear Stack	2-layer Linear Network: Linear(i=128, o=64) + SELU Linear(i=64, o=32) $\times$ 2 (in parallel)
Decoder Convolutional Stack	4-layer Transposed Convolutional Network: TrConv(i=32, o=128, k=4, s=2, p=1) SELU TrConv(i=128, o=64, k=4, s=2, p=1) SELU TrConv(i=64, o=32, k=4, s=2, p=1) SELU TrConv(i=32, o=16, k=4, s=2, p=1)
Decoder Output Layer	Linear(i=16, o=62)

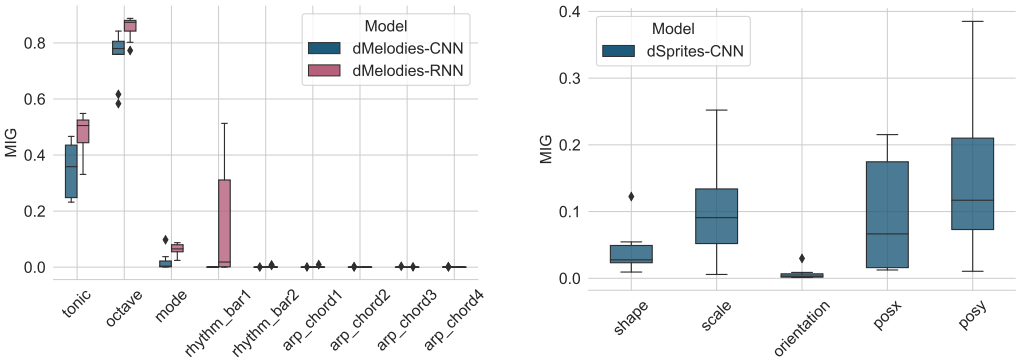
C.3 Additional Results for Supervised Disentanglement

Additional results showing how the regularized attribute of the generated data changes during traversals along the corresponding regularized dimensions for different supervised methods are shown in Figures C.2, C.3, C.4.

Additional results for latent space visualizations (see Section 6.4.7) using encoder distribution plots and decoder latent surface plots are for the different supervised methods are shown below in Figures C.5, C.6, and C.7, respectively.

Table C.2: dSprites-CNN model architecture details. In the Encoder Linear Stack, the last layer has two parallel linear layers for computing the mean and log standard deviation of the latent vectors respectively. Conv: 2-D convolutional layer, TrConv: 2-D transposed convolutional layer, i: input channels, o: output channels, k: kernel size, s: stride, p: padding, ReLU: Rectifier Linear Unit.

Model Type	dSprites-CNN
Encoder Convolutional Stack	4-layer Convolutional Network: Conv(i=1, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU Conv(i=32, o=32, k=4, s=2, p=1) + ReLU
Encoder Linear Stack	3-layer Linear Network: Linear(i=512, o=256) + ReLU Linear(i=256, o=256) + ReLU Linear(i=256, o=10) $\times$ 2 (in parallel)
Decoder Linear Stack	3-layer Linear Network: Linear(i=10, o=256) + ReLU Linear(i=256, o=256) + ReLU Linear(i=256, o=512) + ReLU
Decoder Convolutional Stack	4-layer Transposed Convolutional Network: TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=32, k=4, s=2, p=1) + ReLU TrConv(i=32, o=1, k=4, s=2, p=1)



(a) Annealed-VAE on dMelodies

(c) Annealed-VAE on dSprites

Figure C.1: Factor-wise MIG for Annealed-VAE on dMelodies and dSprites.

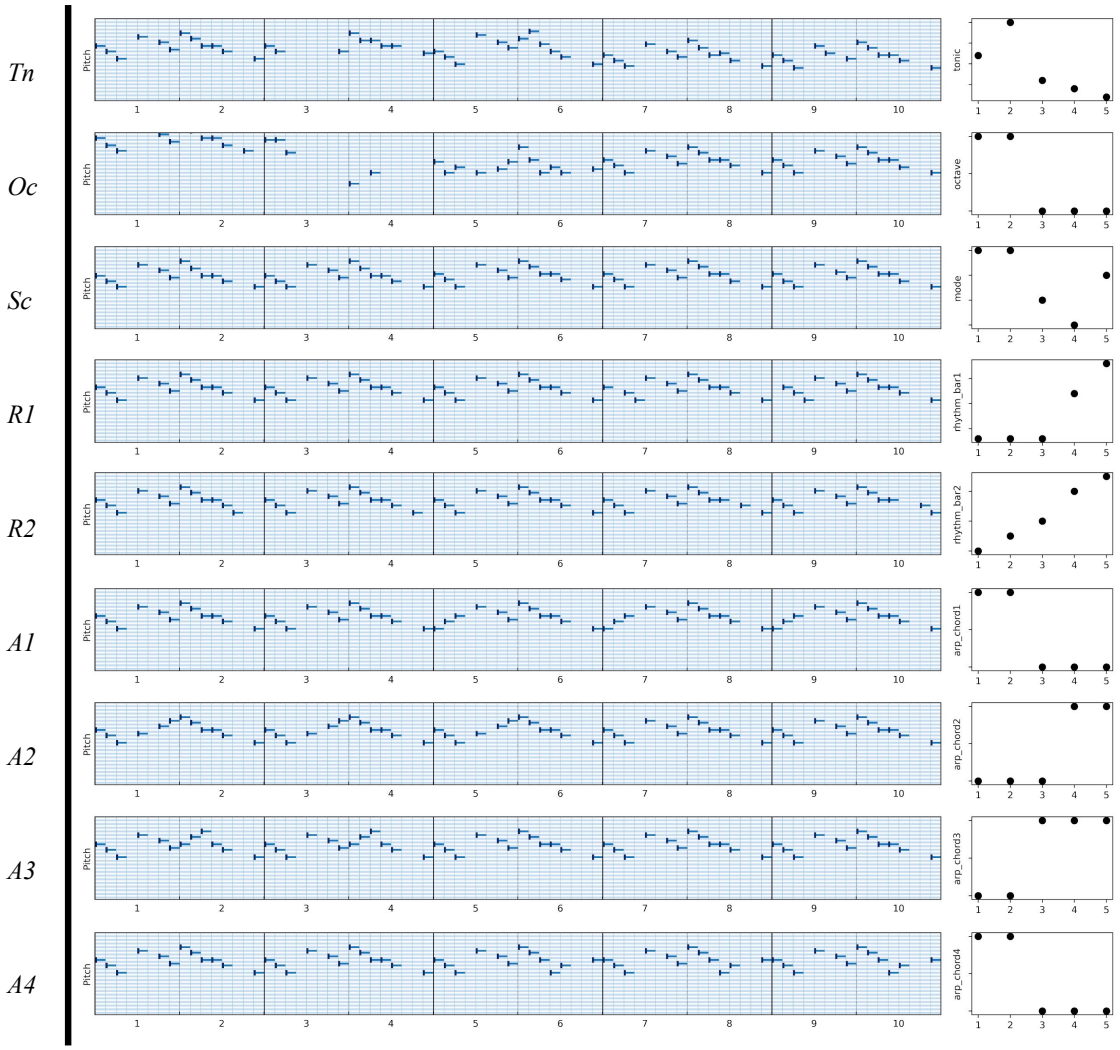


Figure C.2: Attribute manipulation during latent traversal along regularized dimension for the dMelodies-RNN trained using I-VAE. The piano-rolls show measures generated by traversal along the regularized dimension. The plots on the right show how the corresponding attribute values change with the increase in the latent code.

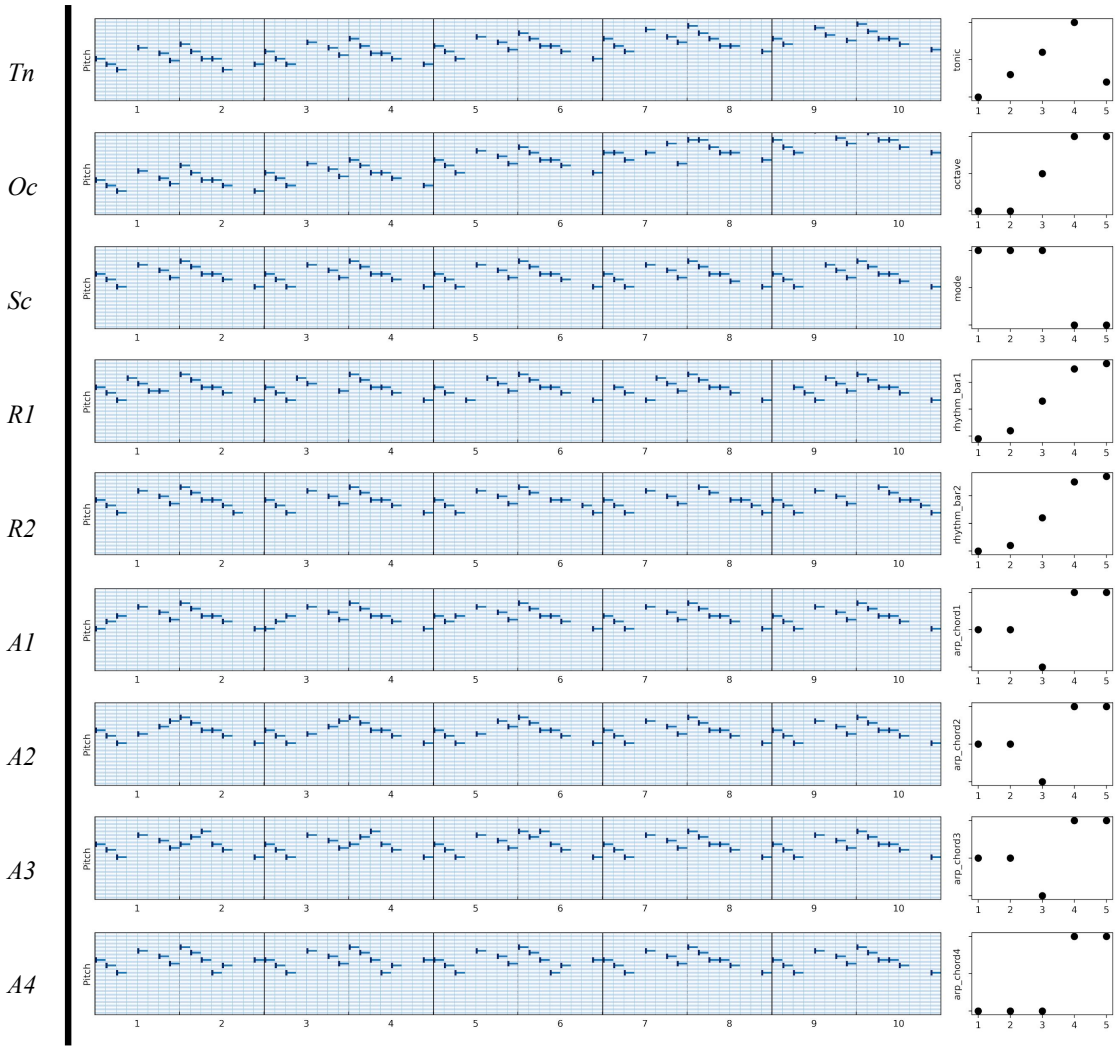


Figure C.3: Attribute manipulation during latent traversal along regularized dimension for the dMelodies-RNN trained using S2-VAE. The piano-rolls show measures generated by traversal along the regularized dimension. The plots on the right show how the corresponding attribute values change with the increase in the latent code.

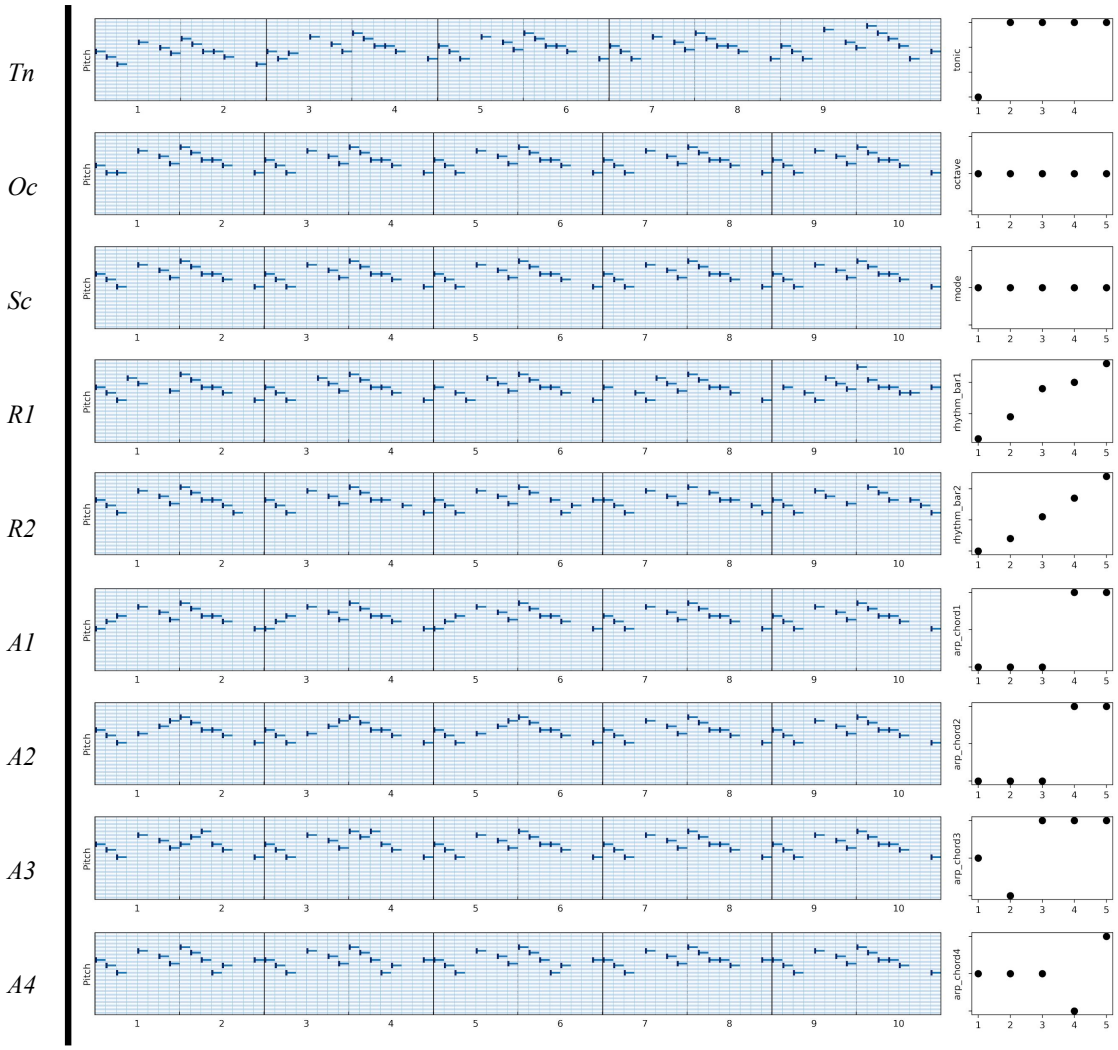
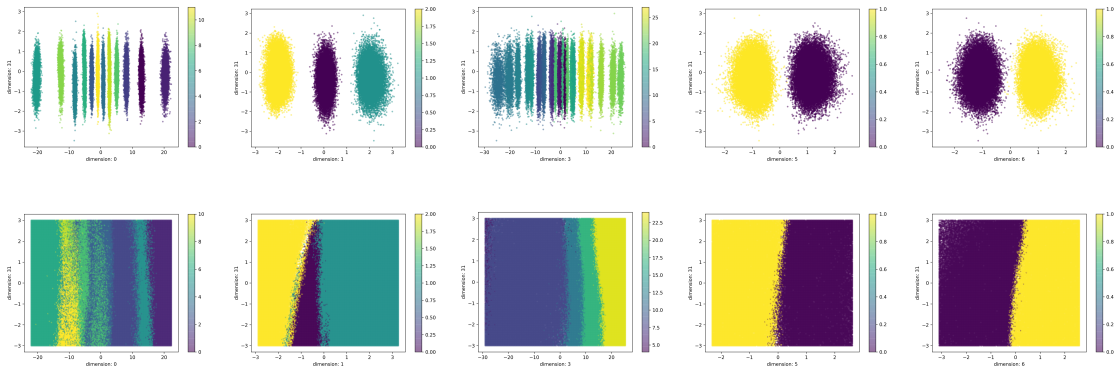
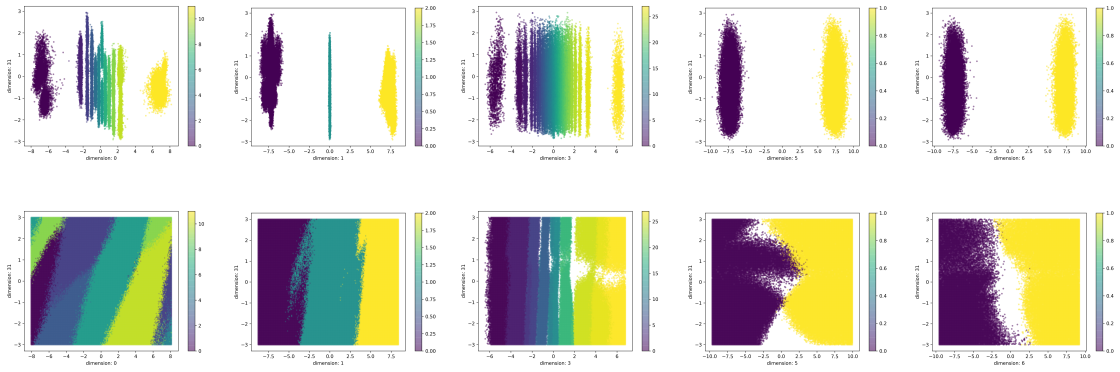


Figure C.4: Attribute manipulation during latent traversal along regularized dimension for the dMelodies-RNN trained using AR-VAE. The piano-rolls show measures generated by traversal along the regularized dimension. The plots on the right show how the corresponding attribute values change with the increase in the latent code.



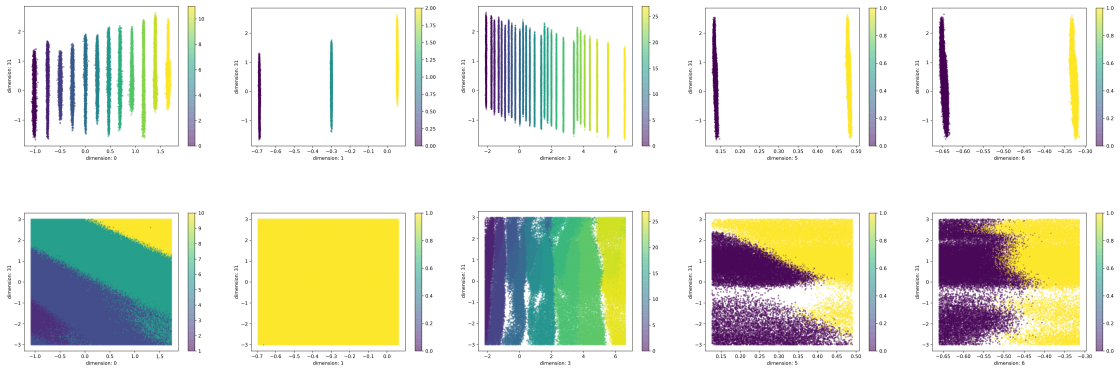
(a) *Tonic* (b) *Octave* (c) *Rhy Bar 1* (d) *Arp Chord 1* (e) *Arp Chord 2*

Figure C.5: Encoder distribution (top row) and surface plots (bottom row) for the I-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.



(a) *Tonic* (b) *Octave* (c) *Rhy Bar 1* (d) *Arp Chord 1* (e) *Arp Chord 2*

Figure C.6: Encoder distribution (top row) and surface plots (bottom row) for the S2-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.



(a) *Tonic* (b) *Octave* (c) *Rhy Bar 1* (d) *Arp Chord 1* (e) *Arp Chord 2*

Figure C.7: Encoder distribution (top row) and surface plots (bottom row) for the AR-VAE method on dMelodies. The x-axis corresponds to the regularized dimension for the attribute and the y-axis corresponds to a non-regularized dimension. Empty regions in the bottom row surface plots denote undefined or out-of-distribution attribute values.

REFERENCES

[1] M. A. Boden, "Creativity," in *Artificial intelligence*, Elsevier, 1996, pp. 267–291.

[2] S. Colton and G. A. Wiggins, "Computational creativity: The final frontier?" In *Proc. of 20th European Conference on Artificial Interlligence (ECAI)*, Montpelier, vol. 12, 2012, pp. 21–26.

[3] A. Cardoso, T. Veale, and G. A. Wiggins, "Converging on the divergent: The history (and future) of the international joint workshops in computational creativity," *AI magazine*, vol. 30, no. 3, pp. 15–15, 2009.

[4] R. Loughran and M. O'Neill, "Application domains considered in computational creativity.," in *Proc. of International Conference on Computational Creativity (ICCC)*, 2017, pp. 197–204.

[5] V. J. Konečni, "Does Music Induce Emotion? A Theoretical and Methodological Analysis," *Psychology of Aesthetics, Creativity, and the Arts*, vol. 2, no. 2, p. 115, 2008.

[6] P. N. Juslin, G. T. Barradas, M. Ovsiannikow, J. Limmo, and W. F. Thompson, "Prevalence of emotions, mechanisms, and motives in music listening: A comparison of individualist and collectivist cultures.," *Psychomusicology: Music, Mind, and Brain*, vol. 26, no. 4, p. 293, 2016.

[7] C. L. Krumhansl, "Music: A link between cognition and emotion," *Current directions in psychological science*, vol. 11, no. 2, pp. 45–50, 2002.

[8] R. Fiebrink, B. Caramiaux, R Dean, and A McLean, *The machine learning algorithm as creative musical tool*. Oxford University Press, 2016.

[9] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[10] J.-P. Briot, G. Hadjeres, and F. Pachet, "Deep Learning Techniques for Music Generation-A Survey," *arXiv preprint arXiv:1709.01620*, 2017.

[11] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweeden, 2018.

[12] F. Colombo, S. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner, "Algorithmic composition of melodies with deep recurrent neural networks," in *Proc. of 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, 2016.

[13] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. of 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, Huddersfield, UK, 2016.

[14] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. of 18th International Society of Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 324–331.

[15] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. of 29th International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[16] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *Proc. of International Conference of Learning Representations (ICLR)*, New Orleans, USA, 2019.

[17] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, pp. 1–13, 2018.

[18] J.-P. Briot and F. Pachet, "Deep learning for music generation: Challenges and directions," *Neural Computing and Applications*, 2018.

[19] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013.

[20] S. Carter and M. Nielsen, "Using Artificial Intelligence to Augment Human Intelligence," *Distill*, vol. 2, no. 12, e9, 2017.

[21] H. Hild, J. Feulner, and W. Menzel, "Harmonet: A neural net for harmonizing chorales in the style of js bach," in *Advances in Neural Information Processing Systems 5 (NeurIPS)*, 1992, pp. 267–274.

[22] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using blstm networks," in *Proc. of 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

[23] I. Simon, D. Morris, and S. Basu, "Mysong: Automatic accompaniment generation for vocal melodies," in *Proc. of SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.

[24] E. Handelman, A. Sigler, and D. Donna, "Automatic orchestration for automatic composition," in *Proc. of 8th Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[25] F. Pachet, "A joyful ode to automatic orchestration," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, pp. 1–13, 2016.

[26] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[27] A. K. Hoover, M. P. Rosario, and K. O. Stanley, "Scaffolding for interactively evolving novel drum tracks for existing songs," in *Workshops on Applications of Evolutionary Computation*, Springer, 2008, pp. 412–422.

[28] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," in *Proc. of International Workshop on Musical Metacreation (MUME)*, Salamanca, Spain, 2018.

[29] K. Collins, "An introduction to procedural music in video games," *Contemporary Music Review*, vol. 28, no. 1, pp. 5–15, 2009.

[30] C. Klimmt, D. Possler, N. May, H. Auge, L. Wanjek, and A.-L. Wolf, "Effects of soundtrack music on the video game experience," *Media Psychology*, vol. 22, no. 5, pp. 689–713, 2019.

[31] C. Donahue, I. Simon, and S. Dieleman, "Piano genie," in *Proc. of 24th International Conference on Intelligent User Interfaces (IUI)*, 2019, pp. 160–164.

[32] R. Typke, F. Wiering, and R. C. Veltkamp, "A survey of music information retrieval systems," in *Proc. of 6th International Conference on Music Information Retrieval*, Queen Mary, University of London, 2005, pp. 153–160.

[33] M. Schedl, E. Gómez Gutiérrez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261.*, 2014.

[34] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics.," in *Proc. of 13th*

International Society for Music Information Retrieval Conference (ISMIR), Citeseer, 2012, pp. 403–408.

[35] W. Chen, J. Keast, J. Moody, C. Moriarty, F. Villalobos, V. Winter, X. Zhang, X. Lyu, E. Freeman, J. Wang, *et al.*, "Data usage in mir: History & future recommendations.," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 25–32.

[36] C.-W. Wu and A. Lerch, "From labeled to unlabeled data – on the data challenge in automatic drum transcription," in *Proc. of 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[37] E. Manilow, G. Wichern, and J. Le Roux, "Hierarchical Musical Instrument Separation," in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[38] R. Liu and S. Li, "A review on music source separation," in *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, IEEE, 2009, pp. 343–346.

[39] B. Gowrishankar and N. U. Bhajantri, "An exhaustive review of automatic music transcription techniques: Survey of music transcription techniques," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, IEEE, 2016, pp. 140–152.

[40] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[41] D. Grocke and T. Wigram, *Receptive methods in music therapy: Techniques and clinical applications for music therapy clinicians, educators and students*. Jessica Kingsley Publishers, 2006.

[42] L. Ferreira and J. Whitehead, "Learning to generate music with sentiment.," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 384–390.

[43] K. Monteith, T. R. Martinez, and D. Ventura, "Automatic generation of music for inducing emotive response," in *Proc. of International Conference on Computational Creativity (ICCC)*, 2010, pp. 140–149.

[44] D. Williams, V. J. Hodge, L. Gega, D. Murphy, P. I. Cowling, and A. Drachen, "AI and Automatic Music Generation for Mindfulness," in *AES International Conference on Immersive and Interactive Audio: Creating the Next Dimension of Sound Experience*, York, 2019.

[45] L. Hiller and L. M. Isaacson, "Illiac suite, for string quartet," vol. 30, no. 3, 1957.

[46] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–30, 2017.

[47] K. Ebcioglu, "An expert system for harmonization of chorales in the style of JS Bach," PhD thesis, Buffalo, NY, USA, 1986.

[48] R. Ovans and R. Davison, "An iterative constraint-based expert assistant for music composition," in *Proc. of Biennial Conference-Canadian Society for Computational Studies of Intelligence*, Citeseer, 1992, pp. 76–76.

[49] D. Lidov and J. Gabura, "A melody writing algorithm using a formal language model," *Computers in the Humanities*, vol. 3, pp. 138–48, 1973.

[50] C. Roads, "Artificial intelligence and music," *Computer Music Journal*, vol. 4, no. 2, pp. 13–25, 1980.

[51] D. Cope, "Experiments in musical intelligence (EMI): Non-linear linguistic-based composition," *Interface*, vol. 18, no. 1-2, pp. 117–139, 1989.

[52] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano, "Using machine-learning methods for musical style modeling," *Computer*, vol. 36, no. 10, pp. 73–80, 2003.

[53] D. Conklin and I. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[54] F. Pachet and P. Roy, "Non-conformant harmonization: the Real Book in the style of Take 6," in *Proc. of 5th International Conference on Computational Creativity (ICCC)*, 2014, pp. 100–107.

[55] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet, "Maximum entropy model for melodic patterns," in *Proc. of ICML Workshop on Constructive Machine Learning*, Lille, France, 2015.

[56] A. Papadopoulos, P. Roy, and F. Pachet, "Avoiding plagiarism in markov sequence generation.," in *AAAI*, 2014, pp. 2731–2737.

[57] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.

[58] D. Eck and J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks," in *12th IEEE Workshop on Neural Networks for Signal Processing*, IEEE, 2002, pp. 747–756.

[59] ——, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, p. 48, 2002.

[60] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: A steerable model for Bach chorales generation," in *Proc. of 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1362–1371.

[61] H Yang and A Hugill, "The creative turn: New challenges for computing," *International Journal of Creative Computing*, vol. 1, no. 1, pp. 4–19, 2013.

[62] B. Genchel, A. Pati, and A. Lerch, "Explicitly conditioned melody generation: A case study with interdependent rnns," in *Proc. of 7th International Workshop on Musical Metacreation (MUME)*, Charlotte, NC, USA, 2019.

[63] D. Makris, M. Kaliakatsos-Papakostas, I. Karydis, and K. L. Kermanidis, "Combining lstm and feed forward neural networks for conditional rhythm composition," in *International Conference on Engineering Applications of Neural Networks*, Springer, 2017, pp. 570–582.

[64] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, *dSprites: Disentanglement testing Sprites dataset*, https://github.com/deepmind/dsprites-dataset, last accessed, 2nd April 2020, 2017.

[65] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26 (NeurIPS)*, Lake Tahoe, USA, 2013, pp. 3111–3119.

[66] J. Engel, M. Hoffman, and A. Roberts, "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models," in *Proc. of 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[67] G. Hadjeres, F. Nielsen, and F. Pachet, "GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures," in *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI)*, Hawaii, USA, 2017, pp. 1–7.

[68] A. Pati and A. Lerch, "Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders," *Neural Computing and Applications*, 2020.

[69] A. Pati, S. Gururani, and A. Lerch, "dMelodies: A Music Dataset for Disentanglement Learning," in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[70] A. Pati, A. Lerch, and G. Hadjeres, "Learning to Traverse Latent Spaces for Musical Score Inpainting," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[71] A. Pati and A. Lerch, "Latent space regularization for explicit control of musical attributes," in *Proc. of ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract*, Long Beach, California, USA, 2019.

[72] T. Bazin, A. Pati, and G. Hadjeres, *A Model-Agnostic Web Interface for Interactive Music Composition by Inpainting*, Neural Information Processing Systems (NeurIPS), Demonstration Track, Montréal, Canada, 2018.

[73] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[74] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. of 33rd International Conference on Machine Learning (ICML)*, New York, USA, 2016, pp. 1747–1756.

[75] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proc. of 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27 (NeurIPS)*, 2014, pp. 2672–2680.

[77] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. of 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 1530–1538.

[78] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with PixelCNN decoders," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016, pp. 4790–4798.

[79] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proc. of 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1587–1596.

[80] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016, pp. 2234–2242.

[81] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems 27 (NeurIPS)*, Montréal, Canada, 2014.

[82] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, Barcelona, Spain, 2016, pp. 2172–2180.

[83] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep Convolutional Inverse Graphics Network," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015, pp. 2539–2547.

[84] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. of 25th International Conference on Machine Learning (ICML)*, Helsinki, Finland, 2008, pp. 1096–1103.

[85] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[86] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *Proc. of 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[87] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," in *Proc. of 20th Conference on Computational Language Processing*, Taipei, Taiwan, 2015.

[88] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016, pp. 2172–2180.

[89] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations," in *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, California, USA, 2019.

[90] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in beta-VAE," in *NIPS Workshop on Learning Disentangled Representations*, Long Beach, California, USA, 2017.

[91] P. Rubenstein, B. Scholkopf, and I. Tolstikhin, "Learning Disentangled Representations with Wasserstein Auto-Encoders," in *Proc. of 6th International Conference on Learning Representations (ICLR), Workshop Track*, Vancouver, Canada, 2018.

[92] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating Sources of Disentanglement in Variational Autoencoders," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018.

[93] H. Kim and A. Mnih, "Disentangling by Factorising," in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweeden, 2018.

[94] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee, "Weakly-supervised Disentangling with Recurrent Transformations for 3D View Synthesis," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015, pp. 1099–1107.

[95] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep Visual Analogy-Making," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015, pp. 1252–1260.

[96] C. Donahue, Z. C. Lipton, A. Balsubramani, and J. McAuley, "Semantically Decomposing the Latent Spaces of Generative Adversarial Networks," in *Proc. of 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[97] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, Montréal, Canada, 2015.

[98] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2Image: Conditional Image Generation from Visual Attributes," in *Proc. of European Conference for Computer Vision (ECCV)*, Amsterdam, The Netherlands, 2016, pp. 776–791.

[99] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv:1411.1784 [cs, stat]*, 2014.

[100] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader Networks:Manipulating Images by Sliding Attributes," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 5967–5976.

[101] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations," in *Proc. of 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018.

[102] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variations using few labels," in *Proc. of 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[103] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[104] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7064–7073.

[105] T. Adel, Z. Ghahramani, and A. Weller, "Discovering Interpretable Representations for Both Deep Generative and Discriminative Models," in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweeden, 2018, pp. 50–59.

[106] M. Bretan, G. Weinberg, and L. Heck, "A unit selection methodology for music generation using deep neural networks," in *Proc. of 8th International Conference on Computational Creativity (ICCC)*, Atlanta, USA, 2016.

[107] S. Lattner, M. Grachten, and G. Widmer, "A predictive model for music based on learned interval representations," in *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 26–33.

[108] A. Arzt and S. Lattner, "Audio-to-score alignment using transposition-invariant features," in *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 592–599.

[109] M. Bretan and L. P. Heck, "Self-supervised methods for learning semantic similarity in music," in *Proc. of 20th International Society for Music Information Retrieval Conference, (ISMIR)*, Delft, The Netherlands, 2019, pp. 446–453.

[110] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," in *Proc. of 2nd Workshop on Machine Learning for Creativity and Design*, Montréal, Québec, 2018.

[111] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer," in *Proc. of 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[112] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[113] K. Ridgeway and M. C. Mozer, "Learning Deep Disentangled Embeddings With the F-Statistic Loss," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018, pp. 185–194.

[114] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations," in *Proc. of 5th International Conference of Learning Representations (ICLR)*, Toulon, France, 2017.

[115] C. Eastwood and C. K. I. Williams, "A Framework for the Quantitative Evaluation of Disentangled Representations," in *Proc. of 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[116] G. Hadjeres and F. Nielsen, "Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, 2018.

[117] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. of 27th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[118] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley, "Audio inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, 2012.

[119] Ç. Bilen, A. Ozerov, and P. Pérez, "Audio declipping via nonnegative matrix factorization," in *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2015, pp. 1–5.

[120] C. Laguna and A. Lerch, "An efficient algorithm for clipping detection and declipping audio," in *Proc. of 141st AES Convention*, Los Angeles, USA, 2016.

[121] N. Perraudin, N. Holighaus, P. Majdak, and P. Balazs, "Inpainting of long audio segments with similarity graphs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 6, pp. 1083–1094, 2018.

[122] C. Bilen, A. Ozerov, and P. Perez, "Joint audio inpainting and source separation," in *Proc. of International Conference on Latent Variable Analysis and Signal Separation*, Springer, 2015, pp. 251–258.

[123] I. Toumi and V. Emiya, "Sparse non-local similarity modeling for audio inpainting," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 576–580.

[124] G. Hadjeres, J. Sakellariou, and F. Pachet, "Style imitation and chord invention in polyphonic music with exponential families," *arXiv preprint arXiv:1609.05152*, 2016.

[125] S. Lattner, M. Grachten, and G. Widmer, "Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints," *Journal of Creative Music Systems*, vol. 2, 1 2018.

[126] A. Roberts, J. Engel, S. Oore, and D. Eck, "Learning Latent Representations of Music to Generate Interactive Musical Palettes," in *Proc. of Intelligent User Interfaces Workshops (IUI)*, Tokyo, Japan, 2018.

[127] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[128] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. of 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 2342–2350.

[129] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017.

[130] M. S. Cuthbert and C. Ariza, "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," in *Proc. of 11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010.

[131] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[132] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[133] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, 2015, pp. 1171–1179.

[134] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. The method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[135] D. R. Hunter *et al.*, "MM algorithms for generalized Bradley-Terry models," *The annals of statistics*, vol. 32, no. 1, pp. 384–406, 2004.

[136] J. E. Ollen, "A criterion-related validity test of selected indicators of musical sophistication using expert ratings," PhD thesis, The Ohio State University, 2006.

[137] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "PianoTree VAE: Structured Representation Learning for Polyphonic Music," in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[138] T. Bazin and G. Hadjeres, "Nonoto: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. of 10th International Conference on Computational Creativity (ICCC)*, UNC Charlotte, NC, USA, 2019.

[139] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017, pp. 5998–6008.

[140] D. C. Castro, J. Tan, B. Kainz, E. Konukoglu, and B. Glocker, "Morpho-MNIST: Quantitative Assessment and Diagnostics for Representation Learning," *Journal of Machine Learning Research 20 (JMLR)*, 2019.

[141] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. of 1st International Conference on Computer Simulation of Musical Creativity (CSMC)*, Huddersfield, UK, 2016.

[142] G. Toussaint, "A Mathematical Analysis of African, Brazilian and Cuban Clave Rhythms," in *Proc. of BRIDGES: Mathematical Connections in Art, Music and Science*, 2002, pp. 157–168.

[143] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[144] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, Barcelona, Spain, 2016, pp. 4743–4751.

[145] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013.

[146] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," in *Proc. of International Conference on Learning Representations (ICLR), Workshop Track*, San Diego, USA, 2015.

[147] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," in *Proc. of International Conference on Learning Representations (ICLR)*, Toulon, France, 2016.

[148] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018, pp. 10 215–10 224.

[149] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017.

[150] W. Hsu, Y. Zhang, R. J. Weiss, Y. Chung, Y. Wang, Y. Wu, and J. R. Glass, "Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*, Brighton, United Kingdom, 2019.

[151] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Montréal, Canada, 2018.

[152] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017.

[153] M. Connor and C. Rozell, "Representing Closed Transformation Paths in Encoded Network Latent Space," in *Proc. of 34th AAAI Conference on Artificial Intelligence*, New York, USA, 2020.

[154] C. Burgess and K. Hyunjik, *3d-shapes Dataset*, https://github.com/deepmind/3d-shapes, last accessed, 2nd April 2020, 2020.

[155] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3D Chairs: Exemplar Part-based 2D-3D Alignment using a Large Dataset of CAD Models," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, USA, 2014, pp. 3762–3769.

[156] M. W. Gondal, M. Wuthrich, D. Miladinovic, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer, "On the transfer of inductive bias

from simulation to the real world: A new disentanglement dataset," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 15 740–15 751.

[157] M. Bretan and L. Heck, "Learning semantic similarity in music via self-supervision," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[158] S. Gururani, A. Lerch, and M. Bretan, "A comparison of music input domains for self-supervised feature learning," in *Proc. of ICML Workshop on Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract*, Long Beach, California, USA, 2019.

[159] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. of 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 141–149.

[160] S. Lattner, M. Dorfler, and A. Arzt, "Learning complex basis functions for invariant representations of audio," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[161] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[162] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," in *Proc. of IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 3730–3738.

[163] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 516–520.

[164] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 6306–6315.

[165] P. Xu, J. C. K. Cheung, and Y. Cao, "On variational learning of controllable representations for text without supervision," in *Proc. of 37th International Conference on Machine Learning (ICML)*, 2020.

[166] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," in *Proc. of 7th*

International Conference on Learning Representations (ICLR), New Orleans, USA, 2019.

[167] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *Proc. of 37th International Conference on Machine Learning (ICML)*, 2020.

[168] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 971–980.

[169] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An Empirical Exploration of Recurrent Network Architectures," in *Proc. of 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015.

[170] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 7511–7522.